Selected domains and lambda calculi

Roberto M. Amadio, Pierre-Louis Curien

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

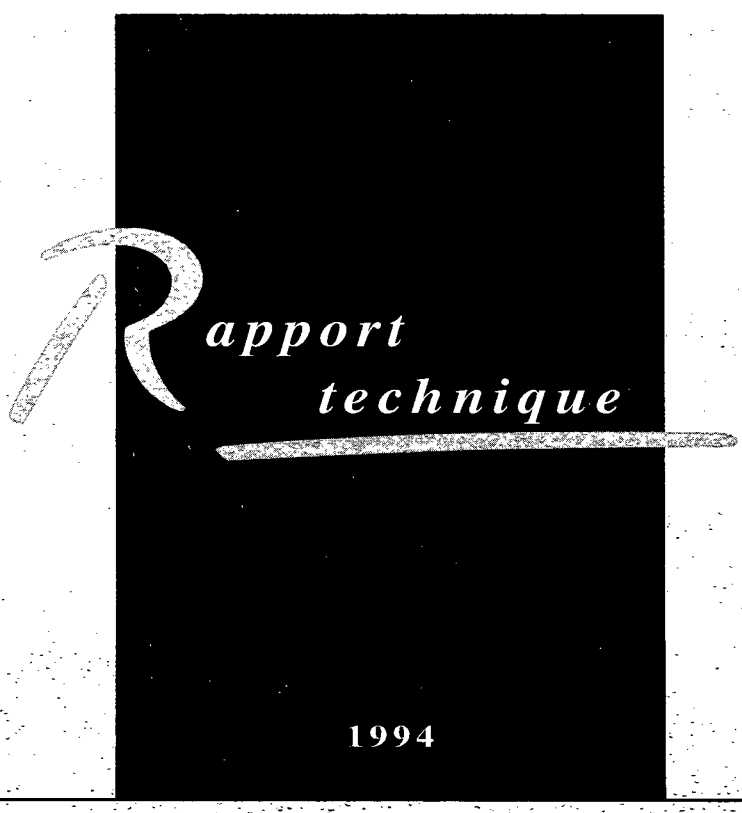
Selected Domains and Lambda Calculi

Roberto M. AMADIO
Pierre-Louis CURIEN

N° 161
Mars 1994

PROGRAMME 2

Calcul Symbolique,
Programmation
et Génie logiciel



Selected Domains and Lambda Calculi

Roberto M. Amadio
CNRS-CRIN-INRIA, Nancy

Pierre-Louis Curien
CNRS-LIENS-DMI, Paris

Une Sélection de Domaines et Lambda Calculs

Résumé

Cette monographie dérive de l'intégration de notes de lecture développées par les auteurs dans les dernières trois années. Les sujets traités incluent: lambda-calcul simplement typé, PCF, interprétations catégorielles, domaines d'ordres partiels complets et algébriques, résultats d'adéquation, lambda-calcul partiel, continuations, domaines puissance, équations aux domaines, types dépendants et du deuxième ordre, sémantique du polymorphisme, stabilité et réalisabilité. Comme support de cours nous présentons deux appendices sur la théorie de la recursion et sur la théorie des catégories.

Abstract

This monograph derives from the integration of lecture notes developed by the authors in the last three years. The topics considered include: simply typed lambda-calculus, PCF, categorical interpretations, domains of algebraic complete partial orders, adequacy results, partial lambda-calculus, continuations, domain equations, dependent and second-order types, semantics of polymorphism, stability, and realizability. As a course support we offer two appendices on recursion theory and category theory.

Contents

Preface

Denotational semantics is concerned with the *mathematical meaning* of programming languages. Programs (procedures, phrases) are to be interpreted in categories with structure (by which we mean sets and functions to start with, and suitable topological spaces and continuous functions to continue with).

The *main goals* of this branch of computer science are, in our belief:

- to provide rigorous definitions that abstract away from implementation details, and that can serve as an implementation independent reference,
- to provide mathematical tools for proving properties of programs: as in logic, semantic models are guides in designing sound proof rules, that can then be used in automated proof-checkers like LCF.

Historically the first goal came first. In the sixties C. Strachey was writing semantic equations in Oxford without knowing if they had mathematical solutions. D. Scott provided the mathematical framework, and advocated its use in a formal proof system called LCF. Thus denotational semantics has from the beginning been applied to the two goals.

In this monograph we aim to present in an elementary and unified way the theory of certain order-theoretic structures that have proved to be useful in the modelling of various families of *typed lambda calculi* considered as core programming languages and as meta-languages for denotational semantics.

The topic is indebted to lattice theory for ideas and techniques, however the aims are different. We look for theories that can be usefully applied to programming languages (and logic). Therefore a certain number of complications arise that are not usually considered. Just to name a few: (a) If we aim to define a model of computation we have to make sure that the transformations are in some sense computable, hence the development of a theory of effective domains and the connections with the theory of enumerated sets. (b) In applications it is difficult to justify the existence of a greatest element, hence the theory is developed without assuming the existence of arbitrary lubs, that is we will not work with complete lattices. (c) There are several models of computation, certainly an important distinction is the possibility of computing in parallel or in series, hence the development of various notions of continuous, stable, and sequential morphisms. (d) There is a distinction between an explicitly typed program and its type-free run time representation, hence the connection with realizability interpretations.

One of our main concerns will be to establish a link between mathematical structures and syntactic problems that arise in the study of programming languages. In our experience it is essential to insist on this part in order to motivate computer scientists to do some mathematics and in order to interest mathematicians in structures that are somehow unfamiliar and far away from the traditional core of mathematics.

A description of the contents of each chapter follows. Unless stated otherwise we do not claim any originality for the material presented here. In this respect we mention in parenthesis the papers which were most influential in the redaction of each chapter, while apologizing for any omission.

Chapter 1 introduces the first basic concepts and, at the same time, gives an idea of some fundamental research lines in domain theory and lambda calculus. Inter alia we discuss: (i) an interpretation of the lambda calculus with a categorical flavour, (ii) a continuation semantics, (iii) the relationship between syntactic and semantic convergence, (iv) the relationship between continuity and calculability (Myhill-Shepherdson theorem), (v) a topological view of domain theory. (Scott[72], Plotkin[83]).

Chapter 2 introduces the paradigms of the interpretation of lambda calculi in categories. In particular we develop the categorical models of simply typed and type free lambda calculus and illustrate the techniques needed to prove the soundness and completeness of the interpretations. (Lambek&Scott[86], Scott[80]).

Chapter 3 gives a complete presentation of the problem of classifying the largest cartesian closed categories of algebraic directed complete partial orders and continuous maps. (Jung[88], Smyth[83]).

Chapter 4 presents an abstract formalization of the notion of computation in terms of monads and applies this idea to three basic types of computations: partial, non-deterministic, and with continuations. We also provide a proof of adequacy for a call by value simply typed λ-calculus. (Moggi[89], Plotkin[85]).

Chapter 5 presents the basic apparatus for the solution of domain equations. It also includes more recent material on the construction of universal homogeneous domains, and on the representation of domains by finitary projections. (Scott[72], Smyth&Plotkin[82], Droste&Göbel[90], Scott[76], Amadio&al.[86]).

Chapter 6 introduces the problem of the categorical interpretation of typed lambda calculi with dependent and second order types along the lines established in chapter 2. We first develop some guidelines in a categorical framework, and then we apply them to the specific cases of categories of complete partial orders and Scott domains. (Girard[86], Coquand&al.[88]).

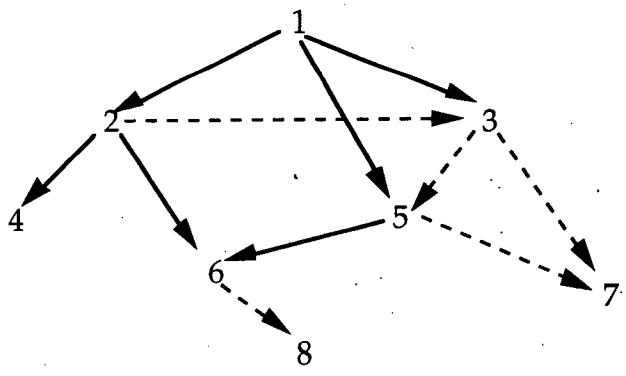
Chapter 7 presents another theory of domains based on the notion of stable map. Stability was first introduced as an approximation of the sequential behaviour of λ-calculus. A stable map is a Scott continuous map which preserves meets of

compatible elements. The chapter develops the theory of algebraic domains and stable maps up to advanced constructions like the one of a retraction of all retractions. (Berry[79], Amadio[91]).

Chapter 8 is an elementary introduction to the ideas of 'synthetic domain theory' via the category of partial equivalence relations (pers). We study some properties of the category of pers up to the point where we can exhibit an interpretation of system F (second order lambda calculus) in this category. Towards the interpretation of recursion we introduce various reflective subcategories of pers, and in this context we prove a generalized Myhill&Shepherdson theorem. (Hyland[88], Rosolini[86], Freyd&al.[90], Amadio[90]).

Two appendices provide the basic material on recursion theory and category theory (see Rogers[67] , Soare[87] for the former and Mac Lane[71], Barr&Wells[85], Asperti&Longo[91] for the latter). For the syntactic aspects of lambda calculus we refer to Barendregt[84], and Girard&al[89].

Most people never manage to read a scientific book from the beginning to the end. We guess this monograph will be no exception, so let us give some advice on how to organize your reading. The following diagram represents strong dependencies (continuous arrows) and weak dependencies (dotted arrows) among the chapters.



When using the monograph in an introductory graduate course or seminar it is perhaps a good idea to modulate the amount of domain theoretic constructions which are presented. For instance in a course whose main emphasis is on lambda-calculi models it is possible to skip chapters 3, 5 (but for the part on domain equations), 7 (but for a reference to the role of stability and sequentiality in the full abstraction problem for PCF), and 8 (but for the interpretation of system F).

This monograph arises out of a joint effort to develop and integrate lecture notes for graduate courses taught by the authors in the years 1991 and 1992 at ENS Paris, ETH Zürich, and CRIN Nancy. The first author is mainly responsible for chapters 2, 4, 5, 6, 7, 8 and the appendix on category theory, while the second author developed chapters 1, 3 and the memento on recursion theory.

Nancy, Paris, Sam 5 Mars 1994

Notation

Pointers:

The monograph is divided in chapters and two appendices on prerequisites in recursion theory and category theory. Each chapter or appendix is divided into sections. Some chapter has an appendix including more technical material. Most important definition, theorems, and exercises are given symbolic names.

Set Theoretic:

- $\cup, \cap, (\bigcup, \bigcap)$: union and intersections of two (a family of) sets.
- X^c is the complement of X.
- P(X) is the parts of X. $P_{fin}(X)$ is the finite parts of X.
- #X is X cardinality.

Category Theoretic:

- **C, D** bold capital letters denote categories.
- **C**[a, b] is the collection of morphisms from a to b.

Domain Theoretic:

- A preorder $(P,\leq)$ is a binary relation on a set that is reflexive and transitive. If the relation is also antisymmetric then we speak of partial order (poset).
- For a subset X of P, we denote by UB(X) the set of upper bounds of X, and by MUB(X) the set of minimal upper bounds (mubs) of X.
- The greatest lower bound, or glb, of a subset X ($\{x,y\}$) of P is written $\sqcap X$ ($x \wedge y$). Similarly, the least upper bound, or lub, of a subset X ($\{x,y\}$) of P is written $\sqcup X$ ($x \vee y$).
- Two elements with a common upper bound are called compatible, and one writes $x \uparrow y$.
- A subset X of a partial order is an upper set if $\forall x\ (x \in X,\ x \leq y \Rightarrow y \in X)$.
- $\uparrow X$ denotes the smallest upper set containing X, i.e. $\uparrow X \triangleq \{y \mid \exists x \in X\ x \leq y\}$. Symmetrically $\downarrow X \triangleq \{y \mid \exists x \in X\ y \leq x\}$.
- An increasing chain (or chain for short) $\{x_n\}_{n \in \omega}$ is a sequence s.t. $x_n \leq x_{n+1}$ for all n.
- If $(P, \leq)$ is a pre-order and $X \subseteq P$ then X is directed if

  $X \neq \emptyset$ and $\forall x,y \in X\ \exists z \in X\ x \leq z \wedge y \leq z$.

Syntax:

- [U/x]V is the substitution of U for x in V, with the standard conventions to avoid capturing of free variables.

Recursion Theoretic:

- $\{n\}$ and ϕ_n are inter-changeable notations for the n-th partial recursive function w.r.t. some given enumeration.

1. Directed Complete Partial Orders

Contents: 1. Assigning Mathematical Meanings to Programs, 2. Domain Equations, 3. Directed Completeness and Algebraicity, 4. Directed Complete Partial Orders as Topological Spaces, 5. Products and Exponentials, 6. Adequacy for PCF, 7. Applications to Proofs.

The plan of this chapter is as follows. We give a quick introduction to denotational semantics by means of a simple toy imperative language, and of a paradigmatic functional programming language, called PCF, that played and still plays an important role in foundational studies (section 1). The interpretation of imperative loops and of recursion operators requires fixpoints of functions. Recursion is also a key issue at the level of data types (section 2). The technical treatment begins with the basic definitions concerning directed complete partial orders (dcpos), complete partial orders (cpos), and algebraic (d)cpos (section 3). Basic connections with topology are hinted at (section 4). Products and function spaces are then analyzed. They introduce the result that the category of dcpos is cartesian closed, as is the full subcategory of cpos, while the full subcategory of algebraic cpos is not (section 5). We shall devote chapter 2 to a careful study of the interpretation of simply typed lambda calculi in cartesian closed categories, and chapter 3 to the various ways of regaining cartesian closure in presence of algebraicity. The results of sections 3 through 5 allow us to complete the denotational semantics of the languages of the first section. The concern in tying operational and denotational semantics receives a precise answer in the form of an adequacy theorem for PCF (section 6). The final section illustrates the use of domains for proving properties of programs (section 7).

1. Assigning Mathematical Meanings to Programs

The basic idea of denotational semantics is to associate with each program piece P a *denotation* $[\![P]\!]$ which is an element of a mathematical structure (a function of some kind). The denotation of a complex construct must be a function of the denotation of the simpler constructs, this feature is often called *compositionality*. At the beginning our mathematical structures will be just *sets*. We shall exemplify this by two languages, the first one is a toy imperative language scheme Imp (taken from Plotkin[83], to which the presentation here is greatly indebted). The second one is the language PCF, a typed λ-calculus with recursion and arithmetic constants, which was also introduced by Plotkin, and has since been used as a paradigmatic language where the relations between denotational and operational semantics can be explored.

An imperative language

We suppose that we are given two unspecified sets BExp and Act of boolean expressions and actions, with generic elements b, a, respectively. The set Stat of statements of the toy imperative language Imp is given by the following syntax:

$s ::= a \mid \text{dummy} \mid s;s \mid \text{if } b \text{ then } s \text{ else } s \mid \text{while } b \text{ do } s$

Let us first leave the while loops apart. Denotational semantics is given abstractly using an unspecified semantic domain (a set for the moment) S of *states* (for example an environment assigning values to identifiers). We write T={**tt**,**ff**} for the set of truth values. To the unspecified syntactic domains BExp and Act correspond unspecified denotation functions $[\![\ ]\!] : \text{BExp} \to S \to T$ and and $[\![\ ]\!] : \text{Act} \to S \to S$ (we use the right associativity convention for $\to$, thus read here $\text{BExp} \to (S \to T)$). Finally the denotation function for statements has type $\text{Stat} \to S \to S$ and is defined by induction on statements as the extension of the semantics of actions such that:

$[\![\text{dummy}]\!] = \text{id}$

$[\![s_1;s_2]\!] = [\![s_2]\!] \circ [\![s_1]\!]$

$[\![\text{if } b \text{ then } s_1 \text{ else } s_2]\!] = \text{condf}([\![b]\!],[\![s_1]\!],[\![s_2]\!])$ where: $\text{condf}(g,h,k)(\sigma) = \begin{cases} h(\sigma) & \text{if } g(\sigma)=tt \\ k(\sigma) & \text{if } g(\sigma)=ff. \end{cases}$

Notice the style of this definition. We could have written the semantics of the conditional statement more simply as:

$$[\![\text{if } b \text{ then } s_1 \text{ else } s_2]\!](\sigma) = \begin{cases} h(\sigma) & \text{if } [\![b]\!](\sigma)=tt \\ k(\sigma) & \text{if } [\![b]\!](\sigma)=ff\,. \end{cases}$$

The first definition is more abstract. Instead of a function, the meaning could be an arrow in a graph (actually a category, since the first and second equation specify the need for identities and composition). Without being so abstract, we shall soon have to turn from mere functions to continuous functions. The first definition directly points to the crucial verifications that will have to be performed to guarantee that this shift can be done: e.g. continuous functions have to be closed under composition, in order to get a continuous meaning for the sequencing of two statements.

The language PCF

We introduce now the language PCF. It is a typed language. The basic entity, or *judgement*, is $\Gamma \supset M:\alpha$, where M is a *term*, α a *type*, and Γ a *typing context*, which is a list of pairs $x:\alpha$ where x is a variable and α is a type. There is a syntax of types and a syntax of terms. There are rules to derive correct judgements $\Gamma \supset M:\alpha$. Not any term is such that $\Gamma \supset M:\alpha$ is provable, for some Γ and α. Terms given by the syntax which follows are called *raw*, terms M in provable $\Gamma \supset M:\alpha$ judgements are called *typable*. The syntax of types, contexts and raw terms is as follows:

$\alpha ::= \text{num} \mid \text{bool} \mid (\alpha \to \alpha)$

$\Gamma ::= \emptyset \mid \Gamma,x:\alpha$ ($\emptyset$ is often denoted just as a blank)

$M ::= \underline{n} \mid succ(M) \mid pred(M) \mid true \mid false \mid zero?(M) \mid$ (for $n \in \omega$)
if M then M else M $\mid$ Y
$x \mid \lambda x:\alpha.M \mid MM$

The typing rules are as follows:

$$\Gamma \supset \underline{n}:num \qquad \Gamma \supset true:bool \qquad \Gamma \supset false:bool$$

$$\frac{\Gamma \supset M:num}{\Gamma \supset succ(M):num} \qquad \frac{\Gamma \supset M:num}{\Gamma \supset pred(M):num} \qquad \frac{\Gamma \supset M:num}{\Gamma \supset zero?(M):bool}$$

$$\frac{\Gamma \supset M:bool \quad \Gamma \supset N:bool \quad \Gamma \supset P:bool}{\Gamma \supset \text{if M then N else P: bool}} \qquad \frac{\Gamma \supset M:bool \quad \Gamma \supset N:num \quad \Gamma \supset P:num}{\Gamma \supset \text{if M then N else P: num}}$$

$$\frac{}{\Gamma,x:\alpha \supset x:\alpha} \qquad \frac{\Gamma \supset x:\alpha \quad (x \neq y)}{\Gamma,y:\beta \supset x:\alpha}$$

$$\frac{\Gamma,x:\alpha \supset M:\beta}{\Gamma \supset \lambda x:\alpha.M:\alpha \to \beta} \qquad \frac{\Gamma \supset M:\alpha \to \beta \quad \Gamma \supset N:\alpha}{\Gamma \supset MN:\beta} \qquad \frac{}{\Gamma \supset Y:((\alpha \to \alpha) \to \alpha)}$$

Remark: Here the context Γ acts as a stack. According to the stated rules, we can have several occurrences of the same variable in Γ. Only the last can be used.

Remark: According to the rules, pred($\underline{0}$) is typable. It would require a more sophisticated type system to reject this term at type checking.

Again, we leave apart Y, and also pred (we do not want to engage in partially defined functions for the moment). We can interpret the rest of the language with sets and functions. The following provisional denotations of types and terms introduce the abstract interpretation of the simply typed lambda calculus in ccc described in chpt. 2.

$[\![bool]\!] = \{tt,ff\}$, $[\![num]\!] = \omega$, $[\![\alpha \to \beta]\!] = [\![\alpha]\!] \to [\![\beta]\!]$ (the set of functions from $[\![\alpha]\!]$ to $[\![\beta]\!]$).

A context Γ is interpreted as the cartesian product of the (interpretations of the) types figuring in it. Formally

$[\![\emptyset]\!]=1$, $[\![\Gamma,x:\alpha]\!]=[\![\Gamma]\!] \times [\![\alpha]\!]$

where 1 is the singleton set {*} which is terminal in the category of sets and functions, i.e. for any set X, there is exactly one function, denoted $!_X$ (! for short) from X to 1 (the constant *).

A judgement $\Gamma \supset M:\alpha$ is interpreted as a function from $[\![\Gamma]\!]$ to $[\![\alpha]\!]$. ρ ranges over elements of $[\![\Gamma]\!]$. We denote a constant function, with say value **a**, by $\mathbf{a} \circ !$. This is a piece of category theory: elements are encoded as arrows from 1 to A, where 1 is the

terminal object, and ! is the unique arrow from $[\![\Gamma]\!]$ to 1. We use bold characters to distinguish succ in the syntax from its meaning **succ** as a function from ω to ω.

$[\![\Gamma \supset \underline{n}:num]\!] = \mathbf{n} \circ !$
$[\![\Gamma \supset true:bool]\!] = \mathbf{tt} \circ !$
$[\![\Gamma \supset false:bool]\!] = \mathbf{ff} \circ !$
$[\![\Gamma \supset succ(M):num]\!] = \mathbf{succ} \circ [\![\Gamma \supset M:num]\!]$
$[\![\Gamma \supset zero?(M):bool]\!] = \mathbf{zero?} \circ [\![\Gamma \supset M:num]\!]$
$[\![\Gamma \supset \text{if } M \text{ then } N \text{ else } P]\!] = condf([\![\Gamma \supset M]\!],[\![\Gamma \supset N]\!],[\![\Gamma \supset P]\!])$ where condf is as above

$[\![\Gamma, x:\alpha \supset x:\alpha]\!] = \pi'$ where $\pi'(\rho,d)=d$
$[\![\Gamma, y:\beta \supset x:\alpha]\!] = [\![\Gamma \supset x:\alpha]\!] \circ \pi$ where $\pi(\rho,d)=\rho$
$[\![\Gamma \supset \lambda x:\alpha.M:\alpha \to \beta]\!] = \Lambda([\![\Gamma, x:\alpha \supset M:\beta]\!])$ where $\Lambda(f)(\rho)(d)=f(\rho,d)$
$[\![\Gamma \supset MN:\beta]\!] = ev \circ \langle [\![\Gamma \supset M:\alpha \to \beta]\!],[\![\Gamma \supset N:\alpha]\!] \rangle$ where $\langle f,g \rangle(\rho)=(f(\rho),g(\rho))$ and $ev(f,a)=f(a)$

We shall often feel free to write $[\![M]\!]$ in place of $[\![\Gamma \supset M:\alpha]\!]$, especially when M is closed, i.e. M has no free variables (then it can be typed w.r.t. the empty context $\emptyset$).

Giving a meaning to each expression is only part of the story. The denotational semantics must also reflect the *operational* semantics, which can be specified by means of *rewriting rules* . We shall see these rules in action in a while. For the moment, we keep in mind that some rules of the form $M \to N$ will be specified. Then the denotational semantics should be such that $[\![M]\!] = [\![N]\!]$ (some authors prefer $[\![M]\!] \leq [\![N]\!]$, giving a dynamic flavour to denotational semantics).

Interpreting recursion

Now we address the interpretation of loops or fixpoints, in the respective languages. Operational intuition suggests that the two following statements have the same behaviour: "while b do a" and "if b then (a ; while b do a) else dummy". Supposing that we shall still be able to interpret the rest of the language as we did above, then $f=[\![\text{while b do a}]\!]$ must satisfy

$condf([\![b]\!], f \circ [\![a]\!], id) = f$

which is a fixpoint equation. Similarly we want to interpret Y by some fixpoint operation, in PCF. Take the following term of PCF, which is easily checked to be typable:

add $\triangleq$ $Y(\lambda f:(num \to num) \to num.\ \lambda x:num.\lambda y:num.$
if zero?(x) then y else succ(f(pred(x))(y))) .

Its meaning should be the addition function, since the term encodes the well known definition of addition by induction in Peano arithmetic. Operational intuition will guide us towards considering *partial* functions instead of total ones, and *continuity* (partiality is needed anyway to interpret pred, since pred(0) is not defined). Suppose we want to compute add $\underline{3}\ \underline{4}$, knowing only about predecessors and successors. We write

$3+4= (2+4)+1 = ((1+4)+1)+1 = ((4+1)+1)+1 = 7$

The reader should actually do this formally, and check that the following rules may be used to obtain $\underline{7}$:

$YM \to M(YM)$	(fixpoint!)
$(\lambda x.M)N \to M[N/x]$	(this is β, the basic rule of λ-calculus)
$succ(\underline{n}) \to \underline{n+1}$	$pred(\underline{n+1}) \to \underline{n}$
$zero?(\underline{0}) \to true$	$zero?(\underline{n+1}) \to false$
if true then M else N $\to$ M	if false then M else N $\to$ N

These rules, applied in leftmost-outermost order (that is, taking the first redex whose first symbol is first met when reading a term from left to right), specify a *deterministic* evaluation strategy (there are other evaluation strategies, as we shall see later).

To perform an addition, one expands *add* as much as needed, using the fixpoint rule. We can give a mathematical counterpart to this notion of expansion. The expression

$\lambda f:(num\to num)\to num.\ \lambda x:num.\ \lambda y:num\ .$
if zero?(x) then y else succ(f(pred(x))(y))

has an unproblematic meaning. Replace everywhere functions by partial functions in the semantic equations given above, interpret pred by

$[\![\Gamma \supset pred(M):num]\!](\rho) = [\![\Gamma \supset M:num]\!](\rho) - 1$

where by convention 0-1 is undefined. The *functional*

$[\![\lambda f:(num\to num)\to num.(\lambda x:num.(\lambda y:num.$
if zero?(x) then y else succ(f(pred(x))(y)))$]\!] \triangleq F$

takes partial functions from ω to ω into partial functions from ω to ω. Call $\perp$ the totally undefined function. Then the following fact is the denotational counterpart of the computation of 3+4 above:

$F^4(\perp)(3)(4)=7$

Let us check this: We can prove successively:

$F(\perp)(x)(y)\downarrow y$ iff $x=0$, $\qquad F^n(\perp)(x)(y)\downarrow x+y$ iff $x<n$.

Notice that all the $F^n(\perp)$ are partially defined functions, and that $F^n(\perp)$ extends $F^m(\perp)$, for any $m<n$. In fact addition is the limit of the sequence $(F^n(\perp))_{n\in\omega}$, in a sense which will be made precise soon. Summarizing, we need fixpoints, and operational intuition suggests us to get them by some limiting process involving partiality, which reflects successive approximations of results during the course of computation. Before we come to the mathematical treatment, we should mention that we may need to go beyond the notion of partial function. Consider the two following terms:

$M=\lambda x:num.\ 0$, $\qquad N=Y(\lambda f:\ num\to num.\ \lambda x:num\ .\ succ(f(x)))$.

Computations of N0 can only loop:

$N0 \rightarrow^* succ(N0) \rightarrow^* succ^n(N0) \rightarrow^* \ldots$

but we get in one step:

$M(N0) \rightarrow 0$ (by β) .

This corresponds to call-by-name: the argument is not evaluated before function application. Suppose that we stay in the framework of sets and partial functions. The meaning of N0 is undefined; what about the meaning of M? We have to capture the idea of a function returning a result even if its argument is undefined. The formal way to do that is to add a special element to ω, called $\perp$. and to turn $\mathbf{N} \triangleq \omega \cup \{\perp\}$ into a partial order by setting

$x \leq y$ iff $x=\perp$ or $x=y$.

This type of partial order is called a *flat* domain. Clearly flat domains are in one-to-one correspondence with sets (morphisms are different, though, see below).

Now, instead of working with partial functions, say from ω to ω, we shall consider the set $\mathbf{N} \rightarrow \mathbf{N}$ of all monotone functions from **N** to **N** (we shall often use $\rightarrow$ without further precision: the context should determine which set of arrows we are speaking of). Notice that f being monotone amounts in this case to:

either $f(\perp)=\perp$ (then f is called *strict*)
or f is constant: $\exists y \in \mathbf{N}\ \forall x \in \mathbf{N}\ f(x)=y$.

This is exactly the extension that was desired above, since there is clearly a bijection between the set $\omega \rightarrow \omega$ of partial functions from ω to ω and the set $\mathbf{N} \rightarrow_{\perp} \mathbf{N}$ of strict functions from **N** to **N** (exercise). If we are interested to model call-by-value only, then staying with partial functions is fine.

Summarizing, our semantic equations have been given in the three following successive frameworks:

- sets and functions (typically, $\omega \rightarrow \omega$)
- sets and partial functions (typically, $\omega \rightharpoonup \omega$) (for the sake of fixpoints)
- partial orders with minimum and monotone functions (typically, $\mathbf{N} \rightarrow \mathbf{N}$) (for the sake of call-by-name).

A digression: continuation semantics

The imperative language which we have introduced lacks an essential feature of imperative programming: **goto**. The effect of this construct is to break the normal sequential flow of a program. We shall now briefly discuss the semantics of an imperative language with gotos, following the notation introduced for our first toy imperative language. The purpose of this exercise is to get the reader acquainted with a more involved view of denotations for programs. The language Imp' which we consider now is:

s ::= a | dummy | s;s | if b then s else s | goto l | l:s

where as before a, b range over predefined sets BExp and Act of boolean expressions and actions, and L ranges over a new predefined set of labels Lab. We call Stat' the set of statements generated by the above syntax. Notice that we have left out "while". The reason is that "while b do s" can be recovered by:

l:(if b then (s;(goto l)) else dummy

as we shall justify in exercise WHILE-GOTO.

The design of a semantics of this language is more involved than for the two previous languages. We have to integrate the goto feature, which complicates the control, or the task of determining "what to do next". The vision we had in our first toy imperative language was that a command acted as a state transformer, and handed its resulting state to the next command. The presence of goto creates a new situation: in the statement s;t, s will be executed while t may possibly not, because s may jump to a completely different area of the program. So it is not of much use if the meaning of s produces a state which t can start with.

An appropriate way of approaching the semantics of "goto" is by means of continuations (the concept was coined by C. Strachey and D. Scott). The main idea is that, since possible jumps make the future, or the *continuation* of the program quite unpredictable, we make 'future' a parameter of the semantics. This guides us to the definition of the following sets:

$C=S\to S$ (C is the set of command continuations, with generic element θ)
$Env=Lab\to C$ (Env is the set of environments with generic element ρ).

The semantic interpretation function for statements, which we shall write $[\![\ ]\!]_c$ (to distinguish it from the function introduced for Imp) will now have the following type:

$[\![\ ]\!]_c : Stat'\to Env\to C\to C$.

The best way to grasp this is to start with the subset Act of Stat', for which the function $[\![\ ]\!]: Act\to S\to S$ is available. The restriction of $[\![\ ]\!]_c$ to Act is defined by:

$[\![a]\!]_c\rho\theta = \theta \circ [\![a]\!]$

The category-theory oriented reader should think of the hom functors $\lambda x.\mathbf{C}(x,a):\mathbf{C}^{op}\to\mathbf{Set}$. We take the semantics of boolean expressions as for Imp, $[\![\ ]\!]$: BExp $\to$ S $\to$ T. We are now ready to describe the extension of $[\![a]\!]_c$ to Stat':

$[\![dummy]\!]_c\ \rho = id$
$[\![s_1;s_2]\!]_c\rho = ([\![s_1]\!]_c\rho) \circ ([\![s_2]\!]_c\rho)$
$[\![\text{if b then } s_1 \text{ else } s_2]\!]_c\rho\theta\sigma = [\![s_1]\!]_c\rho\theta\sigma$ if $[\![b]\!]\sigma=tt$, $[\![s_2]\!]_c\rho\theta\sigma$ if $[\![b]\!]\sigma=ff$
$[\![\text{goto l}]\!]_c\rho\theta = \rho(l)$
$[\![l:s]\!]_c\rho\theta = Fix(\lambda\theta'.[\![s]\!]_c\rho'\theta)$ where $\rho'=\rho[[\![s]\!]_c\rho\theta'/l]$

In the last equation, we have used an updating operation, $\rho[\theta/l]$ is defined by: $\rho[\theta/l](l')=\rho(l')$ for $l'\neq l$, and $\rho[\theta/l](l)=\theta$. Fix is an operator that takes a function as an

argument and computes its (least) fixed point (see sections 2, 3 for the mathematical definition). The fourth equation makes it obvious why we had to take environments as well as continuations as an argument of the semantic interpretation. The following exercise should convince the reader that this semantic setting extends satisfactorily the semantic setting for Imp.

Exercise WHILE-GOTO: Show the following:
$[\![\text{while b do a}]\!]\sigma = [\![\text{l:(if b then (s;(goto l)) else dummy}]\!]_c\perp(\text{id})\sigma$.

2. Domain Equations

When do fixed points of monotone functions exist? An old result is the following:

Proposition *(Fixed points à la Tarski)*
Let D be a complete lattice, i.e. a partial order in which every subset has a glb (equivalently every subset has a lub). Then any monotone map $f:D\to D$ has a least fixed point, which is $\sqcap\{x \mid f(x)\leq x\}$.

Proof
Let $z=\sqcap\{x \mid f(x)\leq x\}$. For any x s.t. $f(x)\leq x$, by definition $z\leq x$, thus by monotonicity $f(z)\leq f(x)$, thus by transitivity $f(z)\leq x$, from which we infer $f(z)\leq z$. By monotonicity again $f(f(z))\leq f(z)$, from which we infer $z\leq f(z)$. Thus z is a fixed point. It is a least fixed point since if $f(z')=z'$, then a fortiori $f(z')\leq z'$. □

Exercise TARSKI-FP: Show by a simple adaptation of the argument of proposition *Fixed points à la Tarski* that the proposition can be strengthened: the set of fixed points of D is a complete lattice (this is the actual Tarski fixed point theorem).

Exercise SCHRÖDER-BERNSTEIN: Let X, Y be sets. Show that if $f:X\to Y$ and $g:Y\to X$ are injections, then there exists a bijection $h:X\to Y$.

This proposition may give the impression that we could stay with monotone functions. But we are forced one step further - to continuity, for a number of reasons:
(1) Tarski's least fixed point gives no handle on how the least fixed point can actually be effectively computed.
(2) It requires a complete lattice, thus in particular a maximum element $\top$. Authors differ in the appreciation of how much an inconvenient this is. A lot of denotational semantics (e.g. Scott[72], Scott[76]) was done in a framework of complete lattices. Exercise ANTITOP gives reasons against $\top$.
(3) An important motivation for continuity is a theorem of recursion theory, due to Rice and Shapiro, which asserts that "computable implies continuous", in a sense which is made precise in the next section.

(4) We do not only have to give meaning to recursive definitions of programs, but also to recursive definitions of domains. We cannot solve these equations in the category **Set** of sets and functions, for reasons of cardinality.

Proposition *(Cantor's diagonalization)*

For any set D with more than one element, there is no surjection from D onto $D \to D$.

Proof

Suppose there is one, call it ϕ. Then by the axiom of choice, we can build $f: D \to D$ s.t. $f(d) \neq \phi_d(d)$ (where like in recursive function theory we write ϕ_d for $\phi(d)$). Suppose there is an e s.t. $f = \phi_e$. Then in particular $f(e) = \phi_e(e)$, contradicting the specification of f. Note: the same result holds in the category **Pord** of partial orders and monotone functions, but it is harder to prove (see Dilworth&Gleason[62]). □

Dynamic binding

The semantics of the following basic declarative language based on dynamic binding will provide an illustration of the use of domain equations in semantics.

M ::= n | x | let x bedyn M in M.

The intended value of

let x bedyn 3 in (let y bedyn x) in let x bedyn 5 in y

is 5, in contrast with the λ-calculus discipline, which is static (the corresponding λ-term is $(\lambda x.(\lambda y.(\lambda x.y)5)x)3$ which evaluates to 3).

The operational semantics can be described via rewrite rules on pairs (M,σ), written $M[\sigma]$, until eventually a constant n is reached. M ranges over terms, σ ranges over *syntactic* environments consisting of a finite set of pairs (M/x); all the x's are distinct. We set $\sigma(x)=M$ if M/x belongs to σ, and $\sigma(x)=x$ otherwise. We denote by $\sigma[N/x]$ the environment τ s.t. $\tau(y)=\sigma(y)$ for any $y \neq x$, and $\tau(x)=N$. The rules are as follows:

$n[\sigma] \to n$,
$x[\sigma] \to \sigma(x)[\sigma]$,
$(\text{let x bedyn M in N})[\sigma] \to N[\sigma[M/x]]$.

A denotational semantics of this language can be given with the help of a semantic domain Env for environments satisfying the equation $\text{Env} = \text{Ide} \to (\text{Env} \to \mathbf{N})$, where Ide is a domain of identifiers (the set of identifiers, with a $\perp$ added, like in **N**, to cope with continuity). The meaning $[\![M]\!]$ of a term M is given as a mapping from Env to **N**.

$[\![n]\!](\rho) = n$,
$[\![x]\!](\rho) = \rho(x)(\rho)$,
$[\![\text{let x bedyn M in N}]\!](\rho) = [\![N]\!](\rho[[\![M]\!]/x])$.

These semantic equations look "the same" as the rules defining the operational

semantics. It is however a non-trivial problem to show the following so-called *adequacy* property of the denotational semantics with respect to the operational semantics:

If M is a closed term, then $M[] \to^* n$ iff $[\![M]\!](\perp) = n$

where M[ ] is the pair of a term M and of the empty syntactic environment. We discuss this in more detail, following Mulmuley[87]. We first need to formulate adequacy for any term. We define a second semantic mapping from syntactic environments to semantic environments in the following way:

$[\![\sigma]\!](x) \triangleq [\![\sigma(x)]\!]$.

The general adequacy result that we want to prove is:

$(M[\sigma] \to^* n \Leftrightarrow [\![M]\!]([\![\sigma]\!]) = n)$, for any M,σ.

The ⇒ direction is rather easy, by induction on the length of the derivation of M[σ] to n:

- n[σ]. We have $[\![n]\!]([\![\sigma]\!])=n$ by the first semantic equation.
- x[σ]: We have by induction $[\![\sigma(x)]\!]([\![\sigma]\!])=n$, and $[\![x]\!]([\![\sigma]\!])= [\![\sigma(x)]\!]([\![\sigma]\!])$ by the second semantic equation and the definition of $[\![\sigma]\!]$.
- (let x bedyn M in N)[σ]: We have by induction $[\![N]\!]([\![\sigma[M/x]]\!])=n$, and $[\![$let x bedyn M in $N]\!]([\![\sigma]\!]) = [\![N]\!]([\![\sigma]\!][[\![M]\!]/x]) = [\![N]\!]([\![\sigma[M/x]]\!])$ by the second semantic equation and the definition of $[\![\sigma]\!]$.

We show how to prove the converse, *if we can prove the existence* of the two following mutually recursively defined predicates $\Theta \subseteq (Env \to \mathbf{N}) \times Exp$ (Exp is the set of expressions of our language) and $\Pi \subseteq Env \times (Ide \to Exp)$:

$\Theta \triangleq \{(d,M) \mid \forall (\rho,\sigma) \in \Pi\ d(\rho)=\perp$ or $M[\sigma] \to^* n$ and $n=d(\rho)\}$
$\Pi \triangleq \{(\rho,\sigma) \mid \forall I\ (\rho(I),\sigma(I)) \in \Theta\}$

We prove $([\![M]\!],M) \in \Theta$ by induction on the size of M.

- n: We have always $[\![n]\!](\rho)=n$, and $n[\sigma] \to n$, for *any* ρ,σ, by the first semantic equation and the first rule.
- x: We have $[\![x]\!](\rho) = \rho(x)(\rho)$ and $x[\sigma] \to \sigma(x)[\sigma]$. Since $(\rho,\sigma) \in \Pi$, we have $(\rho(x),\sigma(x)) \in \Theta$. Then by definition of Θ, if $\rho(x)(\rho) \neq \perp$, then

$(\rho(x),\sigma(x)) \in \Theta$, $(\rho,\sigma) \in \Pi \Rightarrow \sigma(x)[\sigma] \to^* \rho(x)(\rho)$, i.e. $x[\sigma] \to^* [\![x]\!](\rho)$.

- let x bedyn M in N: We have $[\![$let x bedyn M in $N]\!](\rho) = [\![N]\!](\rho[[\![M]\!]/x])$ and (let x bedyn M in N)[σ] → N[σ[M/x]]. Since by induction $([\![M]\!],M) \in \Theta$, then $(\rho[[\![M]\!]/x],\sigma[M/x]) \in \Pi$ by definition of Π. We now exploit $([\![N]\!],N) \in \Theta$, which we also know by induction. If $[\![N]\!](\rho[[\![M]\!]/x]) \neq \perp$, then

$([\![N]\!],N) \in \Theta$, $(\rho[[\![M]\!]/x],\sigma[M/x]) \in \Pi \Rightarrow N[\sigma[M/x]] \to^* [\![N]\!](\rho[[\![M]\!]/x])$,

that is: (let x bedyn M in N)[σ] →* $[\![$let x bedyn M in $N]\!](\rho)$. □

We shall explain in chapter 5 how to solve recursive domain equations, and to

show that the domain Env is well-defined. We shall consider, and prove completely another adequacy statement in section 6 and chapter 4.

Exercise ANTITOP: There are two possible extensions of cond to $\{\bot,tt,ff,\top\}$:

$cond(\top,x,y) = x \vee y$, $\quad cond(\top,x,y) = \top$.

Show that each forces to distinguish statements which are operationally equivalent (i.e. which both terminate with the same value or both do not terminate). Specifically, show that the first extension distinguishes

if b then (if b then e_0 else e_1) else (if b then e_2 else e_3) from
(if b then e_0 else e_3) ,

while the second extension distinguishes

if b_0 then (if b_1 then e_0 else e_1) else (if b_1 then e_2 else e_3) from
if b_1 then (if b_0 then e_0 else e_2) else (if b_0 then e_1 else e_3). □

3. Directed Completeness and Algebraicity

We have already considered sets / functions, sets / partial functions; partial orders / monotone functions. Each of those frameworks form a category. We have motivated continuity in the last sections. Here are the relevant definitions for a simple category of continuous functions. We advise the reader unfamiliar with the vocabulary of category theory to refer when needed to the appendix on categories.

We recall that a subset X of a partial order D is *directed* if $X \neq \emptyset$ and $\forall x, y \in X. \exists z \in X.(x \leq z \wedge y \leq z)$. Non-decreasing sequences form directed sets. A bounded set is not necessarily directed (exercise).

Definition *(Dcpo)*

The category **Dcpo** of *directed complete partial orders* and *continuous* maps has: (i) as objects posets that are directed complete, i.e. s.t. every directed set has a lub. (ii) as morphisms monotone maps that preserve the lubs of directed sets, i.e.

Δ directed $\Rightarrow f(\sqcup\Delta) = \sqcup_{x \in \Delta} f(x)$.

If a dcpo has a minimum element, we call it a complete partial order, or cpo. Intuitively, directed sets are those sets for which a notion of convergence makes sense.

Exercise: (1) A monotone function maps directed sets to directed sets. (2) **Dcpo** is indeed a category, and has **Cpo** as a full subcategory (whose objects are the cpos). (3) Show that the identity functions are continuous, and that the composition of two continuous functions is continuous.

The principal interest of the notion of cpo (or of ω-cpo, see exercise ω-CPO) is that it allows for an "effective" fixpoint construction (substance for effectivity may be found in exercise KL-FIXP)

Proposition *(Fixpoints à la Kleene)*

Let D be a cpo, and f be a continuous endofunction of D. $\sqcup_{n\in\omega} f^n(\bot)$ is a fixpoint of f and is the least prefixpoint of f, where x prefixpoint means $f(x)\leq x$.

Proof

From $\bot\leq f(\bot)$, we get by monotonicity that $\bot=f^0(\bot)$, $f(\bot)=f^1(\bot)$,..., $f^n(\bot)$, ... is an increasing chain, thus is directed. By continuity of f, we have

$$f(\sqcup_{n\in\omega} f^n(\bot))=\sqcup_{n\in\omega} f^{n+1}(\bot)=\sqcup_{n\in\omega} f^n(\bot)$$

since $f^0(\bot)\leq\sqcup_{n\in\omega} f^{n+1}(\bot)$. Suppose $f(x)\leq x$. We show by induction:$\forall n\ f^n(\bot)\leq x$. The basis is clear by minimality of ⊥. Suppose $f^n(\bot)\leq x$: by monotonicity $f^{n+1}(\bot)\leq f(x)$; conclude by transitivity. □

Remark: A fortiori $\sqcup_{n\in\omega} f^n(\bot)$ is the least fixpoint of f. Notice that the Tarski and Kleene fixpoint theorems do not imply each other: there are monotone and non continuous functions between complete lattices, and there are cpos which are not complete lattices.

We shall see later that the fixpoint functional is continuous (see exercise CONT-FIXP). Next we concentrate on certain special elements of a dcpo, which we call compact.

Definition *(Compact Elements)*

Given a dcpo D and $d\in D$ we say that d is *compact* (or isolated) if for each directed Δ:

$$d\leq\sqcup\Delta \Rightarrow \exists x\in\Delta\ (d\leq x).$$

We denote with D_o the collection of compact elements of D.

Exercise: The lub of two compact elements, if any, is compact.

Exercise BICOMPLETE: A partial order $(D,\leq)$ is *bicomplete* if both D and $D^{op}=(D,\geq)$ are directed complete. Let D be a bicomplete dcpo, and A be an arbitrary subset of A. Show that A has a complete set of minimal upper bounds, i.e. for any upper bound z of A, there exists a minimal element in $\downarrow z \cap UB(A)$.

In order to recover a reasonable notion of computability, we will require that each element can be seen as the lub of a directed collection of compacts. In particular, if this collection is countable, there will be a way to connect classical recursion theory to computability over these abstract structures.

Definition *(Algebraic Dcpo)*

A dcpo D is *algebraic* if for all d in D the set $\{d' \in D_o \mid d' \leq d\}$ is directed, and $d = \bigsqcup\{d' \in D_o \mid d' \leq d\}$. Moreover it is an ω-algebraic dcpo if it is algebraic and D_o is denumerable. We denote by **Adcpo** (**ω-Adcpo**) the full subcategory of **Dcpo** consisting of algebraic (ω- algebraic) dcpos. We also write **Acpo** = **Cpo** ∩ **Adcpo**, **ω-Acpo** = **Cpo** ∩ **ω-Adcpo**.

Exercise ALG-∃Δ: The following is a small variant of the definition of an algebraic dcpo. Show that a dcpo D is algebraic iff each element x is the lub of a directed set Δ_x of compact elements. Show moreover that if an assignment of such a Δ_x is given for any x, then $\bigcup\{\Delta_x \mid x \in D\} = D_o$.

Exercise: ω-CPO. There is an alternative, more "elementary" presentation of ω-algebraic dcpos. Call ω-dcpo a partial order such that each (denumerable) increasing sequence has a lub. Show that a partial order is an ω-algebraic dcpo iff it is an ω-dcpo in which any element is the lub of a non decreasing sequence of compact elements.

Algebraic dcpos are determined by the partial order on their subset of compact elements. We describe now a construction which associates with any (pre)order an algebraic cpo. The idea is the following: a (pre)ordered set P can be seen as a collection of partially ordered data. Computations over these data are nothing but directed sets. Actually a directed set is a representative for a computation, as we incline to see two cofinal[1] directed sets as defining the same computation. If we add a condition of downward closure, we get a canonical representative of each computation, and we arrive at the notion of ideal.

Definition *(Ideal)*

Given a preorder $(P, \leq)$, an ideal I is a directed, downward closed subset of P, i.e.:

$\varnothing \neq I \subseteq P$, $\forall x \in I.\ (\downarrow x \subseteq I)$, $\forall x, y \in I.\ \exists z \in I.\ (x, y \leq z)$.

We denote with Ide(P) the collection of ideals over P ordered by set-theoretic inclusion. We say that an ideal I is *principal* if $\exists x \in P.\ (\downarrow x = I)$.

Proposition *(Ideal completion)*

(1) If P is a pre-order, then Ide(P) is an algebraic dcpo whose compact elements are exactly the principal ideals.

(2) Vice versa if D is an algebraic dcpo D, then D and $Ide(D_o)$ are isomorphic in **Dcpo**.

Proof

(1) Let Δ be a directed set of ideals. Define ∪Δ as the set-theoretic union of the ideals in Δ. It is easily checked that this is an ideal, thus it is the lub of Δ in Ide(P). Next we show that $\{\downarrow x \mid \downarrow x \subseteq I\}$ is directed. Indeed, if $\downarrow x \subseteq I$ and $\downarrow y \subseteq I$, then by

[1] X and Y are called cofinal if $\forall x \in X\ \exists y \in Y\ x \leq y$, and $\forall y \in Y\ \exists x \in X\ y \leq x$.

directedness of I, $z \in I$ for some upper bound of x,y. By downward closure of I, $\downarrow z \subseteq I$, thus $\{\downarrow x \mid \downarrow x \subseteq I\}$ is directed. Obviously $I = \bigcup\{\downarrow x \mid \downarrow x \subseteq I\}$. To check that the compact elements are the principal ideals, it is enough, by exercise ALG-$\exists\Delta$, to check that principal ideals are compact, which is obvious.

(2) The inverse maps are $x \mapsto \{d \mid d \leq x\}$, $I \mapsto \bigcup I$. □

Proposition *(Ideal completion as free construction)*

Ideal completion is left adjoint to the forgetful functor U:**Dcpo**$\rightarrow$**P**.

Proof

Take a monotone $f: X \rightarrow D$. The unique continuous extension of g to Ide(X) is defined by $g(\downarrow x) = f(x)$ (extension means: $g \circ \subseteq = f$, where $\subseteq: X \rightarrow Ide(X)$ is the map defined by $\subseteq(x) = \downarrow x$. □

The topologically oriented reader may look at exercise IDE-POINTS for an alternative view of ideal completion. Algebraicity allows us to present the notion of continuous function more constructively.

Proposition ε-δ *(ε-δ continuity)*

If D and E are algebraic, then $f: D \rightarrow E$ is continuous iff it is monotone, and for each $e \in E_o$, $x \in D$ s.t. $e \leq f(x)$, there exists $d \leq x$ s.t. $d \in D_o$ and $e \leq f(d)$.

Proof

Suppose f is ε-δ continuous. Let Δ be directed. We have to show $f(\bigsqcup\Delta) \leq \bigsqcup f(\Delta)$. It is enough to show that for any compact $e \leq f(\bigsqcup\Delta)$, there exists $\delta \in \Delta$ s.t. $e \leq f(\delta)$. Indeed, by ε-δ continuity there exists $d \leq \bigsqcup\Delta$ s.t. $d \in D_o$ and $e \leq f(d)$. Now notice that by compactness of d, $d \leq \delta$ for some $\delta \in \Delta$. Conversely, if f is continuous and $e \leq f(x)$, take a directed $\Delta \subseteq D_o$ s.t. $x = \bigsqcup\Delta$. Then by continuity we can rephrase $e \leq f(x)$ as $e \leq \bigsqcup f(\Delta)$, and by compactness of e we get d s.t. $d \in \Delta$, thus $d \leq x$, and $e \leq f(d)$. □

Exercise ⇔-ε-δ: Prove that under the same assumptions, f is continuous iff

$$e \leq f(x) \Leftrightarrow \exists d \in D_o.(d \leq x \wedge e \leq f(d)).$$

Corollary

Under the assumptions of proposition ε-δ, $\{(d,e) \in D_o \times E_o \mid e \leq f(d)\}$ determines entirely f.

Proof

$f(x) = \bigsqcup\{e \mid e \leq f(x)\} = \bigsqcup\{e \mid \exists d.(d \leq x \wedge e \leq f(d))\}$. □

This gives us a handle to define a notion of effectively continuous function.

Definition *(Effective continuity)*[2]

If D and E are ω-algebraic, and if two surjective enumerations $\{d_n\}_{n\in\omega}$, $\{e_n\}_{n\in\omega}$ of the compact elements of D and E are given, then f: D→E is called effectively continuous iff it is continuous and the set $\{(m,n) \mid e_n \leq f(d_m)\}$ is recursively enumerable.

In the rest of the chapter, we shall not always need algebraicity. We shall explicitly point out when we need it. We provide some examples of dcpos.

(1) Finite partial orders are directed complete. More generally, partial orders which have no infinite ascending chains are directed complete. This follows from the following

Fact FΔM

Let D be a partial order which has no infinite ascending chain. Then
- any directed subset Δ of D has a maximum,
- any monotone function from D to a dcpo D' is continuous.

Proof

Let Δ be a directed set, and take $a\in\Delta$. If a is not maximum, take $b\in\Delta$ s.t. $\neg(b\leq a)$. By directedness, there exists an $a_1 \geq a,b$ in Δ. Hence $a_1 > a$. One may construct this way an increasing chain which must eventually stop with a maximum. Let f:D→D' be monotone. Let Δ be directed in D and δ be its maximum. Then $f(\sqcup\Delta)=f(\delta)$ is the maximum of f(Δ). □

In particular flat partial orders are cpos. Besides **N**, other often used flat domains are $\mathbf{O} \triangleq \{\bot,\top\}$ and $\mathbf{T} \triangleq \{\bot,\mathbf{tt},\mathbf{ff}\}$.

(2) Partial functions provide our next example. Let X⇀Y be the set of partial functions from X to Y, where X and Y are sets. We define

$f\leq g \Leftrightarrow_\Delta \forall x,y\ (f(x)\downarrow y) \Rightarrow (g(x)\downarrow y)$.

In other words f≤g iff the graph of f is included in the graph of g.

Exercise PARTIAL-CPO: Show that X⇀Y is a cpo. If X and Y are countable sets, show that X⇀Y is ω-algebraic, and that its compact elements are the functions which have a finite domain (we call these functions finite).

(3) Our next example is very close to ω⇀ω. But we take only those partial functions whose domain of definition is of the form {0,...,n-1} for some n, or the whole of ω The order is the order inherited from ω⇀ω. This domain can be alternatively described as the union $\omega^*\cup\omega^\omega$ of the sets ω^* of finite and ω^ω of infinite sequences of natural numbers, ordered by the prefix ordering.

[2]Effectively continuous functions are more often called computable continuous functions. Our terminology allows for a less confusing formulation of Myhill-Shepherdson's theorem.

Exercise: Show that $\omega^* \cup \omega^\infty$ is an ω-algebraic cpo, and that its set of non compact elements is $\omega \to \omega$ (in **Set**) (this is known as Baire space). Similarly, one defines the ω-algebraic cpo C of finite or infinite sequences of 0 or 1s. Its set of non compact elements is the Cantor space $\omega \to \{0,1\}$ (in **Set**). For yet another example observe that the powerset of natural numbers, $P\omega$, with set-theoretic containment, is an ω-algebraic cpo.

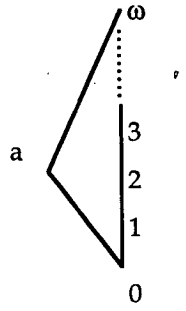
(4) An important instance of the ideal completion construction arises with syntax. Consider a signature Σ consisting of symbols f with an associated arity n=arity(f). Suppose that there is a distinguished symbol Ω of arity 0. Define the following simple theory, establishing judgements $\vdash s \leq t$, where s, t are terms over this signature. It is easy to see that $\{(s,t) \mid \vdash s \leq t \text{ is provable}\}$ is a partial order.

$$\vdash \Omega \leq t$$

$$\frac{\vdash s_1 \leq t_1, \ldots, \vdash s_n \leq t_n}{\vdash f(s_1, \ldots, s_n) \leq f(t_1, \ldots, t_n)}$$

Exercise: Show that the ideal completion of the set of terms is isomorphic to the set of finite and infinite terms, where infinite terms are defined as partial maps S from ω^* to Σ satisfying the following property: $S(un)\downarrow \Rightarrow S(u)\downarrow f$ where n<arity(f).

(5) The following is the minimal example of a non algebraic cpo:



A digression: Computability and continuity

We shall concentrate for a moment on $\omega \to \omega$, and give a recursion-theoretic characterization of the set $(\omega \to \omega) \to_{eff} (\omega \to \omega)$ of effectively continuous functions from $\omega \to \omega$ to $\omega \to \omega$. The reader unfamiliar with recursion theory should skip this at first reading. We recall the Rice-Shapiro theorem (we include a proof of this theorem for convenience in the memento on Recursivity). $(\phi_n)_{n \in \omega}$ is an enumeration of the set PR partial recursive functions. Notice the following inclusions: $(\omega \to \omega)_0 \subseteq PR \subseteq \omega \to \omega$, where $(\omega \to \omega)_0$ consists of the functions which have a finite domain (cf. exercise PARTIAL-CPO).

Theorem RICE-SHAPIRO

Suppose that A is a set of partial recursive functions s.t. $\{n \mid \phi_n \in A\}$ is r.e. Then, for any partial recursive f: $f \in A$ iff there exists a finite function $\theta \subseteq f$ s.t. $\theta \in A$.

Note in particular that A is upward closed. The following theorem makes the motto "computable implies continuous" precise (notice that Rice-Shapiro is already about continuity: "there is a finite function s.t. ...").

Theorem MYHILL-SHEPHERDSON

Suppose that f is a total recursive function which is *extensional*, i.e. $\phi_{f(m)}= \phi_{f(n)}$ whenever $\phi_m= \phi_n$. Then there is a unique continuous function $F:(\omega\rightharpoonup\omega)\rightarrow(\omega\rightharpoonup\omega)$ "extending" f, i.e. s.t. $F(\phi_n)=\phi_{f(n)}$ for all n. Moreover F is effectively continuous. Conversely, any effectively continuous function $F:(\omega\rightharpoonup\omega)\rightarrow(\omega\rightharpoonup\omega)$ maps partial recursive functions to partial recursive functions, and there is a total (extensional) recursive function f s.t. $F(\phi_n)=\phi_{f(n)}$ for all n.

Proof

The first part is proved with the help of Rice-Shapiro. Define $F_o:PR\rightarrow PR$ by $F_o(\phi_n)=\phi_{f(n)}$. The key property of F_o is:

(*) $F_o(f)(m)\downarrow n$ iff $F_o(\theta)(m)\downarrow n$ for some finite $\theta\subseteq f$.

We get this by Rice-Shapiro, taking $A=\{f\in PR \mid F_o(f)(m)\downarrow n\}$ (m,n fixed): a procedure in p which terminates when $\phi_p\in A$ is given by

computing f(p)
then computing $\phi_{f(p)}(m)$ and checking $\phi_{f(p)}(m)=n$.

Since F has to extend F_o, it extends a fortiori the restriction of F_o to finite functions, thus there is no choice for the definition of F but:

$F(f)(m)\downarrow n$ iff $F_o(\theta)(m)\downarrow n$ for some finite $\theta\subseteq f$.

(Hereafter θ always ranges over finite functions.) We show that F is well defined. Suppose that $F_o(\theta)(m)\downarrow n$ and $F_o(\theta')(m)\downarrow n'$ for some finite $\theta,\theta'\subseteq f$. By directedness there is a finite θ'' s.t. $\theta,\theta'\subseteq\theta''\subseteq f$. But, applying (*), one has then $F_o(\theta'')(m)\downarrow n$ and $F_o(\theta'')(m)\downarrow n'$, which forces n=n'.

F extends F_o by definition. It is also continuous by definition (cf. exercise $\Leftrightarrow$-ε-δ). We show finally that F is effectively continuous. A procedure in (encodings of) θ, θ' which terminates when $\theta'\subseteq F(\theta)=F_o(\theta)$ is obtained as a sequence of procedures in θ which terminate when $F_o(\theta)(m)\downarrow n$, for some fixed m,n. Such procedures can be obtained by prefixing to the procedure considered above a (total) procedure taking θ to an index p s.t. $\theta=\phi_p$.

Conversely, let F be effectively continuous. We shall build f as in the statement by a simple application of the s-m-n theorem: it is enough to show that $F(\phi_p)(m)$ is partial recursive in k,m. By a well known characterization of partial recursive functions, this amounts to prove that $F(\phi_p)(m)\downarrow n$ is r.e. in p,m,n. What we know from the effectivity of the continuity of F is that the predicate $F(\theta)(m)\downarrow n$ is r.e. in θ,m,n. Thus we can propose the following procedure (in dovetailing), given p, m

and n:

try in parallel the successive θ s, checking whether $\theta \sqsubseteq \phi_p$ and $F(\theta)(m)\downarrow n$, and stop when one such θ has been found.

Continuity guarantees that the procedure will succeed if $F(\phi_p)(m)\downarrow n$. □

Exercise KL-FIXP: Let F be as in the statement of Myhill-Shepherdson's theorem. Show that the least fixed point of F is in PR.

4. Directed Complete Partial Orders as Topological Spaces

With any partial order $(X,\leq)$ is associated a topology, called *Alexandrov* topology, whose open sets are the upward closed subsets of X. It has as basis the sets $\uparrow x$, where x ranges over X. Conversely, with every topological space $(X,\Omega X)$, one may associate a preorder, called *specialization* preorder, defined by

$$x \leq y \Leftrightarrow_\Delta \forall U \in \Omega X\ (x \in U \Rightarrow y \in U)$$

A T_0 topology is by definition a topology s.t. its associated preorder is a partial order, i.e. if $x \neq y$, then either there exists an open U s.t. $x \in U$ and $y \notin U$, or there exists an open U s.t. $y \in U$ and $x \notin U$. Classical topology considers from the beginning a much stronger separation axiom, known as T_2 or Hausdorff: if $x \neq y$, then there exist disjoint opens U and V s.t. $x \in U$ and $y \in V$. The topological spaces arising from dcpos are *not* Hausdorff (nor are the sober spaces which lead to pointless topology, see Johnstone[82]). They are not even T_1, where T_1 is the following intermediate property: if $x \neq y$, then there exists an open U s.t. $x \in U$ and $y \notin U$.

There is a T_0 topology associated with any dcpo, which is such that
- its specialization order is the dcpo order
- the continuity as defined above order-theoretically coincides with the topological continuity.

The definition of this topology can be arrived at in the following way. Remember that the opens of a topological space X are in one-to-one correspondence with the continuous maps $X \to \{\bot,\top\}$ where the only non trivial open set of $\{\bot,\top\}$ is $\{\top\}$. The specialization order for this topology yields the cpo **O**. So the open sets of a dcpo D must be the sets of the form $f^{-1}(\top)$, for f continuous from D to **O**, in the order-theoretical sense. This should help the reader to find the following definition natural:

Definition *(Scott topology)*
Given a dcpo $(D, \leq)$ let $A \in \tau_S(D)$ iff $A \subseteq D$ and
(i) $x \in A$ and $x \leq y \Rightarrow y \in A$
(ii) Δ directed and $\bigsqcup \Delta \in A \Rightarrow \exists x \in \Delta.(x \in A)$.
The collection $\tau_S(D)$ is a topology, called *Scott* topology.

Exercises (1) U_x-OPEN: Show that $U_x \triangleq \{y \in D \mid \neg(y \leq x)\}$ is open.
(2) SCOTT-TOP: Check that $\tau_S(D)$ is a topology.
(3) NOT-T2: Scott topology is not T_2 unless it is trivial, i.e. $x \leq y$ iff $x = y$.

Now we check that our goals are met.

Fact *(Specialization∘ Scott topology = identity)*
The specialization order on (D,τ_S) is $(D, \leq)$. In particular, τ_S is T_0.
Proof
Call $\leq'$ the specialization order. It is obvious from the definition of Scott topology that $\leq \subseteq \leq'$. Conversely, let $x \leq' y$ and suppose $\neg(x \leq y)$, i.e. $x \in U_y$. Then by definition of $\leq'$ $y \in U_y$, contradicting reflexivity. □

Fact *(Morphisms as Continuous Maps)*
Let D, E be dcpos. The continuous maps, in the topological sense from (D,τ_S) to (E,τ_E) are exactly the morphisms in **Cpo**.
Proof
Let f be τ_S-continuous. Monotonicity of f follows from fact *Specialization∘ Scott topology = identity* (a continuous function is always monotone w.r.t. the specialization order). Suppose $\neg(f(\bigsqcup\Delta) \leq \bigsqcup f(\Delta))$, i.e. $\bigsqcup\Delta \in f^{-1}(U_{\bigsqcup f(\Delta)})$. Thus $f(\delta) \in U_{\bigsqcup f(\Delta)}$ for some $\delta \in \Delta$, since $f^{-1}(U_{\bigsqcup f(\Delta)})$ is open. But this contradicts $f(\delta) \leq \bigsqcup f(\Delta)$. The converse is easy and left to the reader. □

For an algebraic dcpo, Scott topology has a simple basis.

Fact *(Basis of Scott topology)*
If D is algebraic, then the sets $\uparrow d$, d compact, form a basis of τ_S.
Proof
$\uparrow d$ is an open, by definition of compactness. We have to show that if $\uparrow d \cap \uparrow d' \neq \emptyset$, then $\uparrow d'' \subseteq \uparrow d \cap \uparrow d'$, for some d''. Let $x \in \uparrow d \cap \uparrow d'$. Then we find d'' by directedness of $\{e \in D_o \mid e \leq x\}$. We also have to show that if U is open and $x \in U$, then $x \in \uparrow d \subseteq U$ for some d: this is clear by definition of opens and algebraicity. □

Remark: For any dcpo D and $x \in D$, $\uparrow x$ is Scott-open iff x is compact.

The following exercise gives a topological justification to the ideal completion. We

have just recalled that opens are morphisms into **O**. Suppose that we are interested in the dual exercise: we have an abstract topology, i.e. a set of "opens" which is a *frame* A, i.e. a partial order with arbitrary joins and finite meets distributing over them (the set of opens of a topological space, ordered by inclusion, is a frame). Dually to the way of obtaining opens out of points, a way to recover points from these opens is to take the frame morphisms from A to **O** (i.e. those which preserve the frame structure), where **O** is considered as a frame. With this background, the following exercise should appear natural. The construction which takes a topological space to its frame of opens, then to the set of points of this frame, is called soberification.

Exercises (1) IDE-POINTS: Let $(X,\leq)$ be a partial order. Show that ideals of X are in one-to-one correspondence with the points of the Alexandrov topology over X, i.e. the frame homomorphisms from ΩX to **O**.
(2) ACPO-SOB: Show that algebraic dcpos are sober, i.e. they are homeomorphic to their soberification (homeomorphisms are isomorphisms in the category of topological spaces and continuous functions).
(3) Show that the topology induced on $\omega\to\omega$ by the Scott topology (on either $\omega\to\omega$ or $\omega^*\cup\omega^\infty$) is the infinite product of the discrete topologies on the copies of {0,1} (the discrete topology on a set X is **P**(X)).
(4) The set of maximal elements of an algebraic dcpo, equipped with the topology induced by Scott topology, has a clopen base (i.e. a base of sets which are both open and closed).

5. Products and Exponentials

In this section we describe the construction of products and exponentials in the category **Dcpo**. The immediate fall-out of these constructions is that the category is cartesian closed, however we will delay the precise technical meaning of this statement to chapter 2.

We first define products. Let D, E be two dcpos. The product $D\times E$ of D and E in the category of sets becomes a product in the category of dcpos, when endowed with the following componentwise order:

$$(x,y)\leq(x',y') \quad \Leftrightarrow_\Delta \quad x\leq x' \text{ and } y\leq y' .$$

Fact *(Cpo of pairs)*
If D, E are dcpos, then $D\times E$ ordered as above is a dcpo.
Proof
If Δ is directed in $D\times E$, define $\Delta_D = \{x \mid \exists y\ (x,y)\in\Delta\}$, and symmetrically Δ_E. Then $(\sqcup\Delta_D,\sqcup\Delta_E)$ is the lub of Δ. □

Exercise ×- CAT-CPO: This exercise contains the basic ingredients needed to check

that D×E is actually a product of D and E in the category of dcpos. Show:
- The projections π and π' defined respectively by $\pi(x,y)=x$, $\pi'(x,y)=y$ are continuous
- Given f:D→E and g:D→E', $\langle f,g\rangle$ defined by $\langle f,g\rangle(x)=(f(x),g(x))$ is continuous.

Exercise ×-CPO-TOP: If the dcpos are algebraic, the product in **Dcpo** coincides with the product in **Top**, the category of topological spaces. Let D, E be dcpos, and let τ_S,τ be the Scott topology, the product topology on D×E, respectively (a basis of τ is {U×V | U,V open}). Show $\tau\subseteq\tau_S$. Show that if D, E are algebraic, then $\tau=\tau_S$. Can you find a situation where $\tau\neq\tau_S$ (see exercise 1.3.12 in Barendregt[84]) ?

In topology it is not true in general that a function of several arguments is continuous as soon as it is continuous in each argument. This is true for dcpos.

Proposition *(Continuity in each argument is enough)*
Let D, D' and E be dcpos. f:D×D'→E is continuous iff for all x∈D the functions f_x:D'→E, and for all y∈D' the functions f_y:D'→E, defined by $f_x(y)=f(x,y)$ and $f_y(x)=f(x,y)$, respectively, are continuous.
Proof
Let f:D×D'→E be continuous, and Δ be a directed subset of D'. Then $(x,\Delta)\triangleq\{(x,d) \mid \delta\in\Delta\}$ is a directed subset of D×D'. Thus $f_x(\sqcup\Delta) = f(\sqcup(x,\Delta)) = \sqcup f(x,\Delta) = \sqcup f_x(\Delta)$. Suppose conversely that f is continuous in each argument separately. Let Δ be directed in D×D'. Let Δ_D and $\Delta_{D'}$ be as in the proof of Fact *Cpo of pairs*. Then $f(\sqcup\Delta) = f(\sqcup\Delta_D,\sqcup\Delta_{D'}) = \sqcup f(\Delta_D,\sqcup\Delta_{D'}) = \sqcup\{\sqcup f(\delta,\Delta_{D'}) \mid \delta\in\Delta_D\} = \sqcup f(\Delta_D,\Delta_{D'})$. It remains to show $\sqcup f(\Delta_D,\Delta_{D'})=\sqcup f(\Delta)$. One side is obvious since $\Delta\subseteq\Delta_D\times\Delta_{D'}$. Conversely, one uses directedness of Δ to check that each element of $(\Delta_D,\Delta_{D'})$ has an upper bound in Δ. □

Next we consider function spaces.

Fact *(Dcpo of functions)*
Let D, E be dcpos. The set D→E of continuous functions from D to E, endowed with the pointwise ordering defined by $f\leq f' \Leftrightarrow_\Delta \forall x\ f(x)\leq f(x')$, is a dcpo.
Proof
Let Δ be a directed set of functions. Define $f(x)=\sqcup(\Delta(x))$. Let Δ' be a directed subset of D. Then

$$f(\sqcup\Delta')=\sqcup(\Delta(\sqcup\Delta'))=\sqcup\{\sqcup g(\Delta') \mid g\in\Delta\}=\sqcup(\Delta(\Delta'))=\sqcup\{\sqcup(\Delta(\delta')) \mid \delta'\in\Delta'\}=\sqcup f(\Delta').$$ □

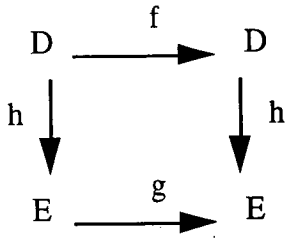
Exercise: →- CAT-CPO: As for products, we collect the basic ingredients needed to show that D→E is a categorical exponential in this exercise. Show:
- The evaluation ev defined by ev (x,y) = x(y) is continuous.
- Given f:D×D'→E show that $\Lambda(f)$:D→(D'→E) defined by $\Lambda(f)(x)(y)=f(x,y)$ is well-defined and continuous.

Going back to fixpoints, we have the following continuity and uniformity results:

Exercise CONT-FIXP: Show that $Fix:(D\to D)\to D$ defined by $Fix(f)=\bigsqcup_{n\in\omega} f^n(\bot)$ is continuous.

This property is important, since it allows to take *parameterized* fixed points (see Proposition *Simultaneous versus Parameterized fixpoints* in section 7).

Exercise UNIF-FIXP: A class of continuous functionals $F_D:(D\to D)\to D$, ranging over all cpos D, is called a fixpoint operator if $F_D(f)$ is a fixpoint of f, for any D and $f:D\to D$. It is called moreover uniform if for every commuting diagram



where h is strict, $h(F_D(f))=F_D(g)$. Show that Fix is the unique uniform fixpoint operator.

The reader may wonder why the structure of cartesian closed category is in order. The reason is that the equations satisfied by cartesian closed categories are used to validate the β-(and η-)rule. This is made precise in the following exercise, which we strongly advise the reader to work out in detail.

Exercise VALIDβ : Show that, for simply typed λ-terms, if $M\to_\beta N$ and $\Gamma\supset M:\alpha$, then $[\![\Gamma\supset M:\alpha]\!] = [\![\Gamma\supset N:\alpha]\!]$ where the semantics is defined as in section 1 (the equations in section 1 make sense in *any* ccc).

What is the situation of *algebraic* dcpos? Unfortunately, if D, E are algebraic, $D\to E$ may not be algebraic (see exercise 0...NONALG). We will see that exponents in **Adcpo** and **Acpo**, if any, must be the exponents in **Dcpo**. Thus **Adcpo** and **Acpo** are not cartesian closed.

The story seems to begin quite well, though, at least in **Cpo**, with the following fact, which shows how compact elements can be naturally constructed in exponents out of compact elements of the components:

Fact *(Step functions)*

Let D, E be cpos, $d\in D$ and $e\in E_o$. Then the *step function* $d\to e$ defined by

$$(d\to e)(x) = e \text{ if } x\geq d, \bot \text{ otherwise}$$

is compact. If D and E are algebraic, then $f=\bigsqcup\{d\to e \mid d\to e\leq f\}$ for any f.

Proof

If $d \to e \leq \bigsqcup \Delta$, then in particular $e=(d \to e)(d) \leq \bigsqcup\{f(d) \mid f \in \Delta\}$. Since e is compact, we get $e \leq f(d)$ for some f, i.e. $d \to e \leq f$. For the second part of the statement, g being an upper bound of $\{d \to e \mid d \to e \leq f\}$ amounts to $e \leq f(d) \Rightarrow e \leq g(d)$, thus $f \leq g$. □

The trouble is that $\{g \mid g \leq f$ and g compact$\}$ need not be directed (see exercise 0...NONALG). It becomes directed if some further axiom is added, as we suggest now. Suppose that $d \to e \leq f$, $d' \to e' \leq f$: if $d \uparrow d'$, then also $e \uparrow e'$. Now, suppose we impose the following axiom:

$x \uparrow y \Rightarrow x \vee y$ exists, for any x and y .

A cpo satisfying this condition is called *bounded complete*. Bounded complete and algebraic cpos are often called *Scott domains*.

Exercise FIN-CONS: Show that an algebraic cpo is bounded complete iff $d \uparrow d' \Rightarrow d \vee d'$ exists, for any *compact* d and d'.

Suppose that E is bounded complete; then define, for $d \to e$ and $d' \to e'$ as above,

$$h(x) \triangleq \begin{cases} e \vee e' & \text{if } d \leq x \text{ and } d' \leq x \\ e & \text{if } d \leq x \text{ and } \neg(d' \leq x) \\ e' & \text{if } \neg(d \leq x) \text{ and } d' \leq x \\ \bot & \text{otherwise .} \end{cases}$$

Exercise: Show that h is the lub of $d \to e$ and $d' \to e'$.

Since finite lubs of compacts are compacts, we get at the same time that $D \to E$ is algebraic and bounded complete. In other words:

Fact *(exponential in Scott-domains)*

If D and E are Scott domains then $D \to E$ is a Scott domain.

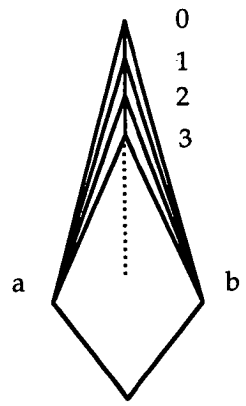
Proof

Call Δ the set of lubs of finite consistent sets of step functions (which always exist by a straightforward extension of the above construction of h). Then $f=\bigsqcup\{d \to e \mid d \to e \leq f\}$ implies $f=\bigsqcup\{g \in \Delta \mid g \leq f\}$, which shows that $D \to E$ is algebraic, since $\{g \in \Delta \mid g \leq f\}$ is directed by definition. Moreover any compact is in Δ, by exercise ALG-$\exists\Delta$. Bounded completeness for finite elements is obvious by definition of Δ. □

But there are larger cartesian closed full subcategories of algebraic dcpos and algebraic cpos, as we shall see in chapter 3.

Exercise 0...NONALG

Show that the following poset D is ω-algebraic, but $D \to D$ is not algebraic.



6. Adequacy for PCF

We have now all the ingredients to come back to the languages introduced in section 1 and complete their denotational semantics. First let us consider the imperative language. We replace everywhere the set S of states by its flat domain extension $S_\perp = S \cup \{\perp\}$, and the set $T \triangleq \{\mathbf{tt},\mathbf{ff}\}$ by its extension $\mathbf{T} \triangleq \{\perp,\mathbf{tt},\mathbf{ff}\}$. We modify the meaning of condf as follows:

$$\mathrm{condf}(g,h,k)(\sigma) \;=\; h(\sigma) \text{ if } g(\sigma)=tt,\;\; k(\sigma) \text{ if } g(\sigma)=ff,\;\; \perp \text{ if } g(\sigma)=\perp.$$

And we add the semantic interpretation of the while loop:

$$[\![\text{while b do s}]\!] = \mathrm{Fix}(F), \text{ where } F(f)=\mathrm{condf}([\![b]\!], f \circ [\![s]\!], Id).$$

In order to show that this semantics is well-defined one has to show, by induction on the size of phrases, that $[\![s]\!]:S_\perp \to S_\perp$ is continuous, for any s (exercise). Of course one now assumes that the predefined meanings of actions and boolean expressions are continuous.

We now turn to the interpretation of PCF. The core λ-calculus kernel of PCF can be interpreted in *any* cartesian closed category. We take here the category **Cpo**. We need to fix the interpretation of basic types and of the constants. Here is the interpretation of types and constants which supersedes (and extends) the provisional one given in section 1:

- $[\![bool]\!] = \mathbf{T}$, $[\![num]\!]=\mathbf{N}$, $[\![\alpha\to\beta]\!]=[\![\alpha]\!]\to[\![\beta]\!]$ (the set of *continuous* functions)
- **succ** is the strict successor function (cf. section 1), and similarly for **zero?** ,
- $[\![pred]\!]$ is the strict and partial predecessor function
- condf is as specified earlier in this section ,
- $[\![Y]\!]=Fix$ (for each type $(\alpha\to\alpha)\to\alpha$ of Y, we consider $Fix:([\![\alpha]\!]\to[\![\alpha]\!])\to[\![\alpha]\!]$).

The validation of the rules other than β of PCF is a straightforward exercise. Thus we have

Proposition *(Soundness of the continuous model of PCF)*

If M,N are PCF raw terms, if $[\![\Gamma \supset M:\alpha]\!]$ and $M \to N$, then N is also typable and $[\![\Gamma \supset M:\alpha]\!] = [\![\Gamma \supset N:\alpha]\!]$.

We have actually concentrated the statement of two properties in this proposition. One is soundness, the other is known as *subject reduction*: if M has type α and M reduces N, then N has also type α.

What about a 'completeness' result of the evaluation w.r.t. the interpretation? There exists such a completeness result for PCF *programs*, which we define as the *closed* terms of *basic* type, e.g. $(\lambda x.x)\underline{3}$ or add $\underline{3}\,\underline{4}$.

Theorem *(Adequacy)*

The continuous model of PCF is *adequate*, i.e. it satisfies, for all programs:

$P \to^* \underline{n}$ for some n $\Leftrightarrow$ $[\![P]\!] = n$

Proof

One implication is an instance of the soundness of the continuous model. The proof of the reverse implication uses the so called *reducibility* (or *computability*, or *realizability*) method, used in various areas of proof theory. The problem is decomposed into two subproblems, one which will be proved by induction on *types*, the other by induction on *terms*. The induction on types come into play by a definition of a family of relations $R_\alpha \subseteq [\![\alpha]\!] \times PCF^\circ_\alpha$, for each type α, where PCF°_α is the set of closed terms of type α. Here is the definition of these relations:

- $R_{num} \triangleq \{(x,P) \mid x=\bot \text{ or } x=n \text{ and } M \to^* \underline{n}\}$
- $R_{\alpha\to\beta} \triangleq \{(f,M) \mid \forall\, (e,N).\ (R_\alpha(e,N) \Rightarrow R_\beta(f(e),MN))\}$

We shall need to prove that these relations satisfy some properties, but we prefer to motivate them first. So we start right away with the second induction of the proof:

Claim 1

For each provable judgement $x_1:\alpha_1,\ldots,x_n:\alpha_n \supset M:\alpha$, for each n-tuple (d_1,N_1), ..., (d_n,N_n) s.t. $R_{\alpha_i}(d_i,N_i)$ for all i's, one has $R_\alpha([\![M]\!](d_1,\ldots,d_n),[N_1/x_1,\ldots,N_n/x_n]M)$, where $[\![M]\!]$ abbreviates $[\![x_1:\alpha_1,\ldots,x_n:\alpha_n \supset M:\alpha]\!]$.

Proof of claim 1

We proceed with the simplest cases first.

- $M=x_i$: Then $[\![M]\!](d_1,\ldots,d_n)=d_i$, and $[N_1/x_1,\ldots,N_n/x_n]M=N_i$, hence the sought result is $R_\alpha(d_i,N_i)$, which is among the assumptions.

- $M=NQ$: Set $M'=[N_1/x_1,\ldots,N_n/x_n]M$, etc... Since by induction $R_{\alpha\to\beta}([\![N]\!](d_1,\ldots,d_n),N')$ and $R_\alpha([\![Q]\!](d_1,\ldots,d_n),Q')$, we get, by definition of $R_{\alpha\to\beta}$:

  $R_\beta(([\![N]\!](d_1,\ldots,d_n))([\![Q]\!](d_1,\ldots,d_n)),N'Q')$, i.e. $R_\beta([\![M]\!](d_1,\ldots,d_n),M')$.

- $M=\lambda x.Q$. Similarly, we set $M'=[N_1/x_1,\ldots,N_n/x_n]M$, $Q'=[N_1/x_1,\ldots,N_n/x_n]Q$. We have

to show, for each pair (d,N) s.t. R(d,N):

$R([\![M]\!](d_1,\ldots,d_n)(d),M'N)$, i.e. $R([\![Q]\!](d_1,\ldots,d_n,d),(\lambda x.Q')N)$.

We have by induction $R([\![Q]\!](d_1,\ldots,d_n,d),[N_1/x_1,\ldots,N_n/x_n,N/x]Q)$. Since $(\lambda x.Q')N \rightarrow [N_1/x_1,\ldots,N_n/x_n,N/x]Q$, we can conclude thanks to condition C1 to be defined below.

- M=$\underline{n}$: Then $R_{num}(n,M)$ holds trivially. Similarly for true and false.

- M=succ(P). We know that $R_{num}([\![P]\!](d_1,\ldots,d_n),P')$ holds. Then either $[\![P]\!](d_1,\ldots,d_n)=\perp$, and then also $[\![M]\!](d_1,\ldots,d_n)=succ(\perp)=\perp$, hence $R_{num}([\![M]\!](d_1,\ldots,d_n),succ(P'))$ holds, or $[\![P]\!](d_1,\ldots,d_n)=n$ for some n, and then $[\![M]\!](d_1,\ldots,d_n)=n+1$, hence $R_{num}([\![M]\!](d_1,\ldots,d_n),succ(P'))$ holds in this case also. Similarly for pred, zero?, cond.

- M=Y. We have to show R(Fix,Y), that is R(Fix(f),YM) for any (f,M) s.t. R(f,M). We need properties C2 and C3 here, to which the reader may wish to jump. Thanks to these properties, we are reduced to show that R(d,YM) implies R(f(d),YM) for any d, which holds by definition of R(f,M) and by property C1, since YM→M(YM). This reasoning is an instance of fixpoint induction, as discussed in the next section. This ends the proof of Claim 1.

We are left with the statement and proof of the following conditions C1, C2, C3:

C1: if R(f,M) and M'→M, then R(f,M')

C2: $R(\perp,M)$

C3: if f_n is a non-decreasing sequence, then $(\forall n\ R(f_n,M)) \Rightarrow R(\sqcup f_n,M)$

Conditions C2 and C3 will be re-examined in the next section.

Claim 2

The relations R_α satisfy the conditions C1, C2 and C3.

Proof of claim 2

C1, C2 are obvious at basic types. For a type $\alpha\rightarrow\beta$, C1 follows by induction from the inference $(M'\rightarrow M) \Rightarrow (M'N\rightarrow MN)$, and C2 follows from the fact that the $\perp$ of $[\![\alpha\rightarrow\beta]\!]$ is the constant $\perp$. C3 follows at basic types from the fact that non-increasing sequences are stationary in a flat domain, and at functional types from the preservation of limits by continuity. This ends the proof of claim 2 and completes the proof of adequacy. □

7. Applications to Proofs

We want to exemplify (for PCF terms) the use of denotational semantics as a means to prove properties of programs. The main tool is *fixpoint* induction. Here is an informal introduction to that proof technique. If we want to show that a property P holds of a term of the form YM, then, knowing that the meaning of YM is the lub of the sequence $\perp$, $F(\perp)$, $F(F(\perp))$,... where F is the meaning of M, it is enough to check the following properties.

- The meaning $[\![P]\!]$ of P is a subdcpo of the domain D associated to the type of YM: in full, $[\![P]\!]$ is closed under limit of non-decreasing chains; we call such a predicate an *inclusive* (or admissible) predicate.
- Both properties $\bot \in [\![P]\!]$ and $\forall x\ (x \in [\![P]\!] \Rightarrow F(x) \in [\![P]\!])$ hold.

This is summarized by the following inference rule, known as *fixpoint induction* principle

$$\frac{Q \text{ inclusive} \ , \ \bot \in Q \ , \ \forall x\ (x \in Q \Rightarrow F(x) \in Q)}{Fix(F) \in Q}$$

where $Q \subseteq D$, for a given cpo D, and $F:D \to D$ is continuous.

Such an inference rule is a step towards mechanizing proofs of programs. What is needed is a formal theory allowing us to prove that some predicates are inclusive. We address this in Exercise INCLUSIVE.

The sufficiency of the above conditions, hence the validity of fixpoint induction is easy to check, using the Peano induction principle on natural numbers. It is not strictly necessary, mathematically speaking, to formulate explicitly the above principle. One can prove $\bot \in [\![P]\!]$, $F(\bot) \in [\![P]\!]$, $F(F(\bot)) \in [\![P]\!]$, and use each time Peano induction to conclude (if P is inclusive). The interest of stating an explicit induction principle is to enable one

- to write lighter proofs: F(x) is easier to write than $F(F(\ldots(\bot)\ldots))$
- to insert it in a mechanical proof-checker like LCF..

We shall carry in some detail the proof of the following proposition, due to Bekic.

Proposition *(Simultaneous versus Parameterized fixpoints)*

Let D,E be cpos and $f:D \times E \to D$, $g:D \times E \to E$ be continuous. Let (x_0,y_0) be the least fixed point of $\langle f,g \rangle$. Let x_1 be the least fixed point of $f \circ \langle Id,h \rangle$, where $h \triangleq Fix \circ \Lambda(g):D \to E$ (Fix is as in Exercise CONT-FIXP. Then $x_0=x_1$ and $y_0=h(x_1)$.

Proof

We shall decompose the problem into the proof of the two inequalities $(x_0,y_0) \leq (x_1,h(x_1))$ and $(x_1,h(x_1)) \leq (x_0,y_0)$.

- Proof of $(x_0,y_0) \leq (x_1,h(x_1))$: Define the predicate Q(u,v) as $(u,v) \leq (x_1,h(x_1))$. This is an inclusive predicate (see Exercise INEQ-INC). Thus we may start the fixpoint induction engine: The basis is obvious. Suppose that $(u,v) \leq (x_1,h(x_1))$. We want to show that $f(u,v) \leq x_1$ and $g(u,v) \leq h(x_1)$. By monotonicity we have $f(u,v) \leq f(x_1,h(x_1))$ and $g(u,v) \leq g(x_1,h(x_1))$. But $f(x_1,h(x_1))=x_1$ since x_1 is a fixed point of $f \circ \langle Id,h \rangle$. This settles the inequality $f(u,v) \leq x_1$. By definition of h, we have $h(x_1)=g(x_1,h(x_1))$, which settles the other inequality.

- Proof of $(x_1,h(x_1)) \leq (x_0,y_0)$: Here we change the predicate to R(u) defined as

$(u,h(u))\leq(x_0,y_0)$. We leave the basis aside for the moment, and suppose that R(u) holds. We have to prove R(f(u,h(u))). We have $f(u,h(u))\leq f(x_0,y_0)=x_0$ by monotonicity, and definition of (x_0,y_0). We need a little more work to obtain $h(f(u,h(u)))\leq y_0$. It is enough to check $h(x_0)\leq y_0$. By definition of h, y_0, we have $h(x_0)=g(x_0,h(x_0))$, and $y_0=g(x_0,y_0)$. That won't do! Nevermind, we take our third inclusive predicate S(u) defined as $u\leq y_0$, remembering that $h(x_0)$ is the least fixed point of $\Lambda(g)(x_0)$. The basis is obvious. Suppose that $u\leq y_0$. Then $g(x_0,u)\leq g(x_0,y_0)=y_0$. Hence fixpoint induction w.r.t. S allows us to conclude $h(x_0)\leq y_0$. We are left with the basis of fixpoint induction w.r.t. R: $(\bot,h(\bot))\leq(x_0,y_0)$ follows a fortiori from $h(x_0)\leq y_0$. □

Let us shortly analize this proof: we have focused in turn on each of the least fixpoint operators involved in the statement, exploiting just the fact that the other least fixpoints are fixpoints.

Exercise INCLUSIVE

- Let D be a cpo. Show that ∅ and D are inclusive predicates on D. Show that $x=x$ and $x\leq y$ are inclusive predicates on $D\times D$.
- Let D and E be cpos and $f:D\to E$ be continuous. Let R be inclusive on E. Show that $f^{-1}(R)$ is inclusive.
- Let D be a cpo and P,Q be inclusive on D. Then show that $P\cap Q$ and $P\cup Q$ are inclusive.
- Let D and E be cpos and P be inclusive on $D\times E$ in its first argument. Show that the predicate $\forall y\ P(x,y)$ is inclusive on D.
- Let D and E be dcpos and P, Q be inclusive in D, E respectively. Show that $P\times Q$ is inclusive in $D\times E$, and that $P\to Q$ is inclusive in $D\to E$, where $P\to Q=\{f:D\to E \mid \forall d\in P\ f(d)\in Q\}$.

Hints for exercises

We do not give hints for the most easy exercises.

WHILE-GOTO: On one hand, we have, setting $U=[\![l:(\text{if b then (s;(goto l)) else dummy}]\!]_c\bot(id)$.

$U\sigma=[\![a]\!]_c\rho'U\sigma$ if $[\![b]\!]\sigma=tt$, σ if $[\![b]\!]\sigma=ff$.

On the other, setting $V=[\![\text{while b do a}]\!]$, we have from section 1:

$V\sigma=V([\![a]\!]\sigma)$ if $[\![b]\!]\sigma=tt$, σ if $[\![b]\!]\sigma=ff$.

Now remember $[\![a]\!]_c\rho'U = U\circ[\![a]\!]$.

TARSKI-FP: Let X be a set of fixed points; the lub of X in $\{x \mid f(x)=x\}$ is $\sqcap\{y \mid f(y)\leq y \text{ and } X\leq y\}$.

BICOMPLETE: Recall that Zorn axiom (which is equivalent to the axiom of choice) asserts that a non-empty partial order has at least a maximal element if

every chain (i.e. a subset for which the induced order is total) has an upper bound. Use Zorn axiom.

CANTOR-SCHRÖDER: Define h as g^{-1} on g(Z), and as f on X-g(Z), where Z is s.t. Z=Y-f(X-g(Z)). Use injectivity of f, which entails, for any subsets A, B: f(A-B)=f(A)-f(B).

ANTITOP: Take $b=\top$, $b_0=\bot$, $b_1=\top$.

ALG-$\exists\Delta$: To show that $\{d'\in D_o \mid d'\leq d\}$ is directed, let $d=\sqcup\Delta$, where Δ is a directed set of compact elements, and notice that if $d'_1\leq d$, $d'_2\leq d$, then by compactness of d'_1, d'_2, there exist $\delta_1, \delta_2\in\Delta$ s.t. $d'_1\leq\delta_1$, $d'_2\leq\delta_2$.

ω-CPO: Suppose D is ω-algebraic. Clearly a cpo is an ω-cpo. Given $x\in D$, let $d_0,\ldots,d_n,\ldots$ be an enumeration of $\{d\in D_o \mid d\leq x\}$. Extract from this sequence an increasing sequence $d'_0,\ldots,d'_n,\ldots$ s.t. $d_n\leq d'_n$. Conversely, with any directed Δ, associate $\{d\in D_0 \mid \exists\delta\in\Delta\ d\leq\delta\}$, enumerate this set and apply the same technique to get a lub for it. Finish by using exercise ALG-$\exists\Delta$.

PARTIAL-CPO: Let Δ be a directed set of partial functions. Then for any a, if $f\in\Delta$ and $a\in$dom(f), then f(a) does not depend on the choice of f. $dom(\sqcup\Delta)=\sqcup\{dom(f) \mid f\in\Delta\}$.

KL-FIXP. Define k by $k(n)=f^n(e_0)$, where e_0 is a code for the $\bot$ function. Then $Fix(F)(x)\downarrow y$ iff $\phi_{k(n)}(x)\downarrow y$.

IDE-POINTS: Given I, define a point p by $p(\uparrow x)=\top$ iff $x\in I$. Given a point p take $I=\{x \mid p(\uparrow x)=\top\}$.

PROD-SUBCAT: Use an equational axiomatization of products. In general take a signature Σ and a set of equations Eq, a Σ-algebra A, a (Σ,Eq)-algebra B and an injective homomorphism $F:A\to B$ of Σ-algebras.

$\times$-CPO-TOP: Remember that the sets $U\times V$ (U,V open) form a basis of the product topology. Let $\sqcup\Delta\in U\times V$: use U,V open and directedness of Δ to obtain $\delta\in\Delta\cap(U\times V)$. Conversely, if W is Scott open in $D\times E$ and $(x,y)\in W$, then $(d,e)\in W$ for some compacts $d\leq x$, $e\leq y$, and $(\uparrow d)\times(\uparrow e)\subseteq W$.

$\to$- CAT-CPO: Use the characterization of proposition *Continuity in each argument is enough.*

CONT-FIXP: The proof is similar to the proof of $\to$CPO.

UNIF-FIXP: Prove by induction $h(f^n(\bot))=g^n(\bot)$ (the basis is by strictness). Use that Fix(f) is the unique fixpoint of f considered as an endofunction of $Fix(f)\downarrow$.

VALIDβ: Check that if $\Gamma,x:\alpha,\Delta\supset M:\beta$ and $\Gamma\vdash N:\alpha$, then
$[\![\ \Gamma,\Delta\supset M:\beta]\!] = [\![\ \Gamma,x:\alpha,\Delta\supset M[N/x]:B]\!] \circ (\langle Id,[\![\Gamma\supset N:\alpha]\!]\rangle\times id)$.

FIN-CONS: Take $x\vee y=\sqcup\{d\vee d' \mid d\leq x,d'\leq y\}$. Notice $\sqcup\{d\vee d' \mid d\leq x,d'\leq y\} = (\sqcup\{d \mid d\leq x\})\vee(\sqcup\{d' \mid d'\leq y\})$.

0...NONALG: Show that the set of approximants of the identity is not directed; specifically show that

$a \to a, b \to b \leq f \leq id \Rightarrow f(\omega) \subseteq \omega$ and $f(d)=d$ if $(d \notin \omega)$

$\Rightarrow \bigsqcup_{n \in \omega} f_n = f$ where $f_n(d) =$ if $(d \notin \omega$ or $d \leq n)$ then $f(d)$ else $f(d+1)$.

Conclude by noticing that $f=f_m$ entails f stationary, contradicting $f \leq id$.

2. Interpretation of Lambda Calculi in Cartesian Closed Categories

Contents: 1. Simply Typed Lambda Calculus. 2. Cartesian Closed Categories. 3. Interpretation of Lambda Calculi in CCC. 4. From CCCs to Lambda Theories. 5. From Lambda Theories to CCCs. 6. Type Free Lambda Calculus and its Interpretation in CCCs. Appendix 1: CCC of Retractions. Appendix 2: Embedding Algebras in Lambda Models.

In first approximation typed lambda calculi are natural deduction presentations of certain fragments of minimal logic. These calculi have a natural computational interpretation as core of typed functional languages where the process of computation ($\beta(\eta)$-reduction) can be seen as the normalization of the proofs corresponding to the terms.

In section 1 we introduce a minimal typed lambda calculus that corresponds to the natural deduction formalization of the implicative fragment of propositional implicative logic. This calculus comes with a notion of $\beta\eta$-reduction and with a corresponding notion of $\beta\eta$-equivalence. As in the case of the recursion free fragment of PCF (chpt.1, sect. 1) it will turn out that simple models can be found by interpreting types as sets and terms as functions between these sets. But, in general, which are the structural properties that characterize such models ? The central problem considered in this chapter is that of understanding what is the "model theory" of such a minimal calculus.

In order to answer this question we introduce in section 2 the notion of cartesian closed category (CCC). We present CCCs as a natural categorical generalization of certain adjunctions found in Heyting algebras. The description of the models of a calculus by means of category theoretic notions will be a central and recurring topic of this monograph. We will not always fully develop the theory but in this chapter we can take advantage of the simplicity of the calculus to go into a complete analysis.

In section 3 we describe the interpretation of the calculus into an arbitrary CCC, and we present some basic properties such as the "substitution lemma". Next, in section 4, we address the problem of understanding which kind of equivalence is induced on terms by such an interpretation. To this purpose we introduce the notion of lambda-theory. Roughly speaking a lambda theory is a congruence over lambda terms that is invariant under $\beta\eta$-equivalence and it turns out that every CCC induces a lambda theory.

We take the view that this is a natural notion to consider, and we ask the

question: does any lambda theory come from the interpretation in a CCC ? In section 5 we answer this question positively by showing how to build a suitable CCC from any lambda theory. We may then consider achieved the program of developing a "model theory" for the simply typed calculus. In section 6 we introduce the type-free λβ-calculus as a typed lambda calculus with a "reflexive type". We show that every CCC with a reflexive object gives rise to a λβ-theory.

We end the chapter with two appendices. The first appendix shows how to build a category of retractions out of a reflexive object in a CCC. This construction can be used to show that *every* lambda theory for the type free lambda calculus is the theory of a reflexive object in a CCC (this is the analogoue of the result presented in section 5 for the simply typed lambda calculus). The second appendix illustrates the coding of algebraic structures in certain lambda-models. This result suggests the richness of lambda-models and the possibility of coding in them standard mathematical structures.

This chapter is mainly based on Lambek&Scott[86], Scott[80], Curien[86] to which the reader seeking more advanced results is addressed.

1. Simply Typed Lambda Calculus

When trying to interpret λ-calculi (or proofs) in structures (categories) it is convenient to have a sequent-style formulation of natural deduction. The reason being that the interpretation depends not only on the term but also on the assumptions (the treatment of assumptions is a weak point of natural deduction formulations as presented, e.g., in Troelstra&VanDalen[88]).

Types and terms are defined by the following BNFs:[3]

Types: $\alpha ::= t \mid s \ldots \mid (\alpha \to \alpha)$

Terms: $M ::= x \mid y \ldots \mid (\lambda x{:}\alpha.M) \mid (MM)$

A *context* Γ is a *list of pairs*, $x{:}\,\alpha$, where x is a variable, all variables are distinct, and α is a type. We write $x{:}\,\alpha \in \Gamma$ to express the fact that the pair $x{:}\,\alpha$ occurs in Γ. If we want to point out the last pair occurring in the list Γ we write $\Gamma, x{:}\,\alpha$. A judgment is a sequent of the shape $\Gamma \supset M{:}\,\alpha$. If a judgment is provable we write $\vdash \Gamma \supset M{:}\,\alpha$. We also write $M{:}\,\alpha$ to denote that there exists a context Γ such that $\vdash \Gamma \supset M{:}\,\alpha$. Provable judgments are inductively defined by the following rules:

(asmp) $\quad x{:}\,\alpha \in \Gamma \Rightarrow \Gamma \supset x{:}\,\alpha$

$(\to I) \quad \Gamma, x{:}\alpha \supset M{:}\beta \Rightarrow \Gamma \supset (\lambda x{:}\alpha.M){:}\,(\alpha \to \beta)$

$(\to E) \quad \Gamma \supset M{:}\,(\alpha \to \beta)\ ,\ \Gamma \supset N{:}\alpha \Rightarrow \Gamma \supset (MN){:}\,\beta$

Exercise SEQvsNAT: First show that the structural rules of exchange, weakening, and contraction are derived in the system above, in the sense that if the premises are provable then the conclusion is provable.

[3] In the following we will feel free to spare on parentheses.

(exch) if $\vdash \Gamma, x{:}\alpha, y{:}\beta, \Gamma' \supset M{:}\gamma$ then $\vdash \Gamma, y{:}\beta, x{:}\alpha, \Gamma' \supset M{:}\gamma$
(weak) if $\vdash \Gamma \supset M{:}\beta$, $x \notin \Gamma$ then $\vdash \Gamma, x{:}\alpha \supset M{:}\beta$
(contr) if $\vdash \Gamma, x{:}\alpha, y{:}\alpha \supset M{:}\beta$ then $\vdash \Gamma, z{:}\alpha \supset [z/x, z/y]M{:}\beta$, for z fresh variable.

Next consider the following natural deduction formulation of the simply typed lambda calculus. There is an infinite collection of *distinct* variables labelled with their type, say x^α, y^β,... The collection of well-typed terms is inductively defined as follows:

$(asmp_N)$ $x^\alpha{:}\,\alpha$
$(\to I_N)$ $M{:}\beta \Rightarrow \lambda x^\alpha.M{:}\,\alpha\to\beta$
$(\to E_N)$ $M{:}\,\alpha\to\beta$, $N{:}\alpha \Rightarrow MN{:}\,\beta$

Define suitable translations between typing proofs and study the equivalence of the two systems.

Reduction Rules: we consider two basic rules for the reduction of terms:

(β) $(\lambda x{:}\alpha.M)N \mapsto [N/x]M$
(η) $\lambda x{:}\alpha.(Mx) \mapsto M$, if $x \notin FV(M)$

we denote by $\to_{\beta\eta}$ their compatible (or contextual) closure, and by $\twoheadrightarrow_{\beta\eta}$ the reflexive and transitive closure of $\to_{\beta\eta}$.

Exercise SUBJ-RED: Well typed terms are closed under reduction, formally:
if $\vdash\Gamma\supset M{:}\,\alpha$ and $M \to_{\beta\eta} N$ then $\vdash\Gamma\supset N{:}\,\alpha$.

Fact *(confluence and normalization)*
We refer to Girard&al.[89] and Bardendregt[84], respectively, for proofs of the following properties:
(1) The reduction system $\to_{\beta\eta}$ is strongly normalizing on well-typed terms, that is if $M{:}\,\alpha$ then all reduction strategies lead to a $\beta\eta$-normal form.
(2) The reduction relation $\twoheadrightarrow_{\beta\eta}$ is confluent, hence each term has a unique $\beta\eta$-normal form. Confluence of $\beta\eta$ is a property independent from the strong normalization, for instance it also holds for the type-free calculus. □

2. Cartesian Closed Categories

The reader will find in the appendix some basic notions of category theory. We now try to motivate the introduction of CCCs as the combination of two more elementary examples.

Example *(Conjunction and Binary Products)*
Let us consider a simple calculus in which we can pair two values or project a pair to one of its components. This fragment corresponds to the natural deduction rules for conjunction in minimal logic.

Types: $\alpha ::= t \mid s... \mid (\alpha \wedge \alpha)$
Terms: $M ::= x \mid y... \mid \langle M, M\rangle \mid (\pi_1\, M) \mid (\pi_2\, M)$

(asmp) $x: \alpha \in \Gamma \Rightarrow \Gamma \supset x: \alpha$
($\wedge$I) $\Gamma \supset M:\alpha\ ,\ \Gamma \supset N:\beta \Rightarrow \Gamma \supset \langle M, N\rangle: \alpha\wedge\beta$
($\wedge$E1) $\Gamma \supset M: \alpha\wedge\beta \Rightarrow \Gamma\supset \pi_1\, M: \alpha$
($\wedge$E2) $\Gamma \supset M: \alpha\wedge\beta \Rightarrow \Gamma\supset \pi_2\, M: \beta$

It is intuitive that a cartesian category (i.e. a category with a terminal object and binary products) has something to do with this calculus. Let us make this intuition more precise:
(1) We suppose to interpret types as objects of a cartesian category **C**. If we denote with A the interpretation of a type α, then the interpretation of the type $\alpha\wedge\beta$ should be the cartesian product $A\times B$.
(2) If types are objects it seems natural to associate terms to morphisms. If M is a closed term of type α we may expect that its interpretation is a morphism $f: 1\to A$, where 1 is the terminal object. But what about a term M s. t.: $\vdash x_1:\alpha_1,...,x_n:\alpha_n \supset M:\alpha$? The idea is to interpret this term as a morphism $f: (..(1\times A_1)\times...\times A_n)\to A$.

This example has now served the goal of suggesting that types can be seen as objects and terms as morphisms. We do not wish to be more precise at the moment (but see section 3) and leave the following as an

Exercise: Define an interpretation of the typed terms of the calculus presented above into a cartesian category. □

Example: We assume that the reader knows of the correspondence between classical propositional logic and boolean algebras. For intuitionistic (or minimal) logic the following structures serve a similar role:

Definition *(Heyting algebras)*
A Heyting algebra, H, is a lattice with join operation "$\vee$", meet operation "$\wedge$", greatest element 1, least element 0 and with a binary operation "$\to$" that satisfies the condition: $x\wedge y \le z$ iff $x \le y \to z$.

Exercise: Heyting algebras abound in nature! Show that the collection of open sets Ω of a topological space (X, Ω) ordered by inclusion can be seen as a Heyting algebra by taking: $U\to V \triangleq \text{Interior}(X\setminus U \cup V) \equiv \bigcup \{W\in\Omega \mid W \subseteq X\setminus U \cup V\}$.

For our purposes the important point in the definition of Heyting algebra is that the implication is characterized by an adjoint situation (in a poset case), namely:

$$\forall y\in H \quad \_\wedge y \dashv y \to \_$$

i.e. for any $y\in H$ the function $y \to \_$ is right adjoint to the function $\_\wedge y$. □

In poset categories the interpretation of proofs is trivial. For this reason Heyting

algebras cannot be directly applied to the problem of interpreting the simply typed $\lambda\beta\eta$-calculus. However combined with our previous example they suggest a natural generalization: consider a cartesian category in which each functor $\_\times A$ has a right adjoint $(\_)^A$. In this way we arrive at the notion of CCC. This condition can be reformulated in a more explicit way, as shown in the following definition.

Definition *(CCC)*

A category **C** is cartesian closed if it has:

(i) A *terminal object* 1.

(ii) For each A, B $\in$ **C** a *product* that we denote with: $A \xleftarrow{\pi^A} A\times B \xrightarrow{\pi^B} B$ such that

$\forall C\in \mathbf{C}.\ \forall f: C\to A.\forall g: C\to B.\exists! h: C\to A\times B.$

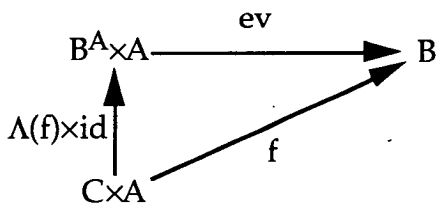
$(\pi^A \circ h = f \wedge \pi^B \circ h = g),$

(h is often denoted by $\langle f, g\rangle$, π_1 and π_2 are abbreviations for π^A and π^B respectively)

(iii) For each A, B $\in$ **C** an *exponent* that we denote with: $ev: B^A\times A \to B$ such that

$\forall C\in \mathbf{C}.\ \forall f: C\times A\to B.\ \exists! h: C\to B^A.\ ev \circ (h\times id) = f,$

(h is often denoted by $\Lambda(f)$, Λ is the *currying* operator).



□

Exercise CCC-Funct: Given a CCC **C** define a product functor Prod: $\mathbf{C}\times\mathbf{C}\to\mathbf{C}$ and an exponent functor Exp: $\mathbf{C}^{op}\times\mathbf{C}\to\mathbf{C}$ such that $Prod(A, B) = A\times B$ and $Exp(A, B) = B^A$.

Exercise CCC-ADJ: Show that a CCC can be characterized as a category **C** such that the following functors have a right adjoint: (i) the unique functor $!: \mathbf{C}\to\mathbf{1}$, (ii) the diagonal functor $\Delta: \mathbf{C}\to\mathbf{C}\times\mathbf{C}$, (iii) the functor $\_\times A: \mathbf{C}\to\mathbf{C}$, for any object A.

Exercise CCC-EQS: Show that a CCC can be characterized as a category **C** such that:

(i) there are: $1\in\mathbf{C}$, and $*_A: A\to 1$, for any $A\in\mathbf{C}$ such that:

$\forall f: A\to 1.\ (f = *_A).$

(ii) there are: $A \xleftarrow{\pi^A} A\times B \xrightarrow{\pi^B} B$, for any A, B$\in$**C**, and $\langle f, g\rangle: C\to A\times B$ for any $f: C\to A$, $g: C\to B$ such that:

(Fst), (Snd) $\forall f: C\to A.\forall g: C\to B.\ (\pi^A \circ \langle f, g\rangle = f \wedge \pi^B \circ \langle f, g\rangle = g)$

(SP) $\forall h: C\to A\times B.\ \langle \pi^A\circ h, \pi^B\circ h\rangle = h$

(iii) there are: $ev: B^A\times A \to B$, for any A, B$\in$**C**, and $\Lambda(f): C\to B^A$ for any $f: C\times A\to B$, such that:

(β_{cat}) $\forall f: C\times A\to B.\ ev \circ (\Lambda(f)\times id) = f$

(η_{cat}) $\forall h: C\to B^A.\ \Lambda(ev \circ (h\times id)) = h$

where as usual: $f\times g \equiv \langle f\circ\pi^A, g\circ\pi^B\rangle$. □

Exercise CCL: Referring to CCC-EQS prove that: (i) (SP) is equivalent to:

(DPair) $\forall f: C\to A.\forall g: C\to B.\ \forall h: D\to C.\ \langle f,g\rangle\circ h = \langle f\circ h, g\circ h\rangle$

(FSI) $\forall A, B\in \mathbf{C}.\ \langle \pi^A, \pi^B\rangle = id$

(ii) (β_{cat}), (η_{cat}) are equivalent to:

(Beta) $\forall f: C\times A\to B.\ \forall g: C\to A.\ ev \circ \langle \Lambda(f), g\rangle = f \circ \langle id, g\rangle$

(DΛ) $\forall f: C\times A\to B.\ \forall h: D\to C.\ \Lambda(f)\circ h = \Lambda(f\circ (h\times id))$

(AI) $\forall A, B\in \mathbf{C}.\ \Lambda(ev) = id$. □

Exercise EX-CCC: Show that the following categories are cartesian closed: (a) (Finite) Sets. (b) (Finite) Posets and Monotone Functions. On the other hand prove that the category pSet of sets and partial functions is not cartesian closed. Hint: consider the existence of an isomorphism between pSet[2×2, 1] and pSet[2, 4].

We can now formally prove that the category of *directed complete partial orders* (dcpos) and maps preserving joins of directed sets is cartesian closed (see chapter 1).

Proposition *(Dcpo is a CCC)*

The category **Dcpo** is cartesian closed.

Proof

The terminal object is the poset with one element. For the product of two dcpos consider the cartesian product ordered componentwise. Verify that projections and pairing are morphisms in the category and satisfy the equations associated to the product. For the exponent E^D just consider the hom-set Dcpo[D, E] ordered pointwise. Verify that evaluation and *currying* are morphisms in the category and that they verify the equations associated to the exponent. □

Exercise CPO-CCC: Verify that also the category **Cpo** is cartesian closed.

Exercise PROD-SUBCAT: Let **C**, **C'** be categories, and $F:\mathbf{C}\to\mathbf{C}'$ be a faithful functor. Suppose that **C'** has products, and that for any pair of objects A and B of **C** there exists an object C and two arrows $\alpha:C\to A$ and $\beta:C\to B$ s.t.

- $F(C)=F(A)\times F(B)$, $F(\alpha)=\pi$ and $F(\beta)=\pi'$,
- for any object D and arrows $f:D\to A$, $g:D\to B$, there exists an arrow $h:D\to C$ s.t. $F(h)=\langle F(f),F(g)\rangle$.

Show that **C** has products. Explain why this general technique applies to **Dcpo** and **Cpo**. Try to formulate a general statement in universal algebra that captures this technique.

Exercise EXP-SUBCAT: Let **C**, **C'** be categories, and $F:\mathbf{C}\to\mathbf{C}'$ be a faithful functor. Suppose that the assumptions of exercise PROD-SUBCAT hold, and use × to

denote the cartesian product in **C**. Suppose that **C'** has exponents, and that for any pair of objects A and B of **C** there exists an object C, a mono $m:F(C)\rightarrowtail(A\rightarrow B)$ and an arrow γ: $C\times A\rightarrow B$ s.t.

- $F(\gamma)=ev \circ (m\times Id)$
- for any object D and arrows $f:D\times A\rightarrow B$, $g:D\rightarrow B$, there exists an arrow $k:D\rightarrow C$ s.t. $m \circ F(k) = \Lambda(F(f))$. Then show that **C** has exponentials. Apply this to **Dcpo**.

3. Interpretation of Lambda Calculi

In this section we explain how to interpret the simply typed lambda calculus introduced in section 1 in an arbitrary CCC. Suppose **C** is a CCC. Let us choose a terminal object 1, a product functor Prod: $\mathbf{C}\times\mathbf{C}\rightarrow\mathbf{C}$ and an exponent functor Exp: $\mathbf{C}^{op}\times\mathbf{C}\rightarrow\mathbf{C}$. Then there is an obvious interpretation for types as objects of the category. The arrow is interpreted as exponentiation in **C**. If α, β are types we denote with A, B their interpretations. We also denote with B^A the exponentiation of B to the A (according to the chosen functor Exp).

Consider a provable judgment of the shape: $x_1: \alpha_1,..., x_n: \alpha_n \supset M: \alpha$. Its interpretation will be defined by *induction the length of the derivation* as a morphism from $1\times A_1\times...\times A_n$ to A. We will take the convention that $\times$ associates to the left. Then we denote with $\pi_{n,i}$: $1\times A_1\times...\times A_n\rightarrow A_i$ $(i=1,...,n)$ the morphism: $\pi_2\circ\pi_1\circ....\circ\pi_1$, where π_1 is iterated (n-i) times.

(asmp) $\quad [\![x_1: \alpha_1,..., x_n: \alpha_n \supset x_i: \alpha_i]\!] = \pi_{n,i}$

($\rightarrow$I) $\quad [\![\Gamma \supset \lambda x{:}\alpha.M{:}\alpha\rightarrow\beta]\!] = \Lambda([\![\Gamma, x{:}\alpha \supset M{:}\beta]\!])$

($\rightarrow$E) $\quad [\![\Gamma\supset MN:\beta]\!] = ev\circ\langle [\![\Gamma \supset M: \alpha\rightarrow\beta]\!] , [\![\Gamma\supset N:\alpha]\!] \rangle$

The last two rules need some explanation. Suppose $\Gamma\equiv x_1: \alpha_1,..., x_n: \alpha_n$, and $C\equiv 1\times A_1\times...\times A_n$. ($\rightarrow$I): if we have a morphism $f: C\times A\rightarrow B$ then we have a uniquely determined morphism $\Lambda(f): C\rightarrow B^A$. ($\rightarrow$E): if we have two morphisms $f: C\rightarrow B^A$ and $g: C\rightarrow A$, then we can build the morphism $\langle f, g\rangle: C\rightarrow B^A\times A$ and composing with ev we get $ev\circ\langle f, g\rangle: C\rightarrow A$.

Notes

(1) The interpretation above is defined by induction on the structure of a proof of a judgment $\Gamma \supset M: \alpha$. In the simple system we presented here a judgment has a unique proof. However, in general, there can be several ways of deriving the same judgment, therefore a problem of *coherence of the interpretation* arise, namely one has to show that different proofs of the same judgment receive the same interpretation. Note that in the simply typed calculus the coherence problem is avoided by getting rid of the structural rules. This trick does not suffice in more sophisticated type theories like LF or the calculus of constructions where the derivation is not completely determined by the structure of the judgment. More precisely the problem comes from the inter-dependence of term judgments and

type equality judgments.

(2) The categorical interpretation can be seen as a way of compiling a language with variables into a language without variables. The slogan is that *variables are substituted by projections*, for instance: $[\![\varnothing \supset \lambda x{:}\alpha.x{:}\alpha{\to}\alpha]\!] = \Lambda(\pi_2)$. In other words rather than giving a symbolic reference in the form of a variable one provides a path for accessing a certain information in the context.[4] As a matter of fact the "compilation" of the lambda calculus into the categorical language has been taken as a starting point for the definition of an abstract environment machine (the Categorical Abstract Machine) in the style of Landin's classical SECD machine (see Curien[86]).

Exercise INT-WEAK: Show that if $\vdash\Gamma \supset M{:}\beta$ and x does not occur in Γ then $\vdash\Gamma,x{:}\alpha \supset M{:}\beta$ (cf. SEQvsNAT) and $[\![\Gamma,x{:}\alpha \supset M{:}\beta]\!] = [\![\Gamma \supset M{:}\beta]\!] \circ \pi_1$.

Exercise INT-EXCH: Given the two contexts $\Gamma, x{:}\alpha, y{:}\beta, \Gamma'$ and $\Gamma, y{:}\beta, x{:}\alpha, \Gamma'$ it is possible to define a natural isomorphism τ between the corresponding objects. For example if $\Gamma \equiv z{:}\gamma$ and $\Gamma' \equiv \varnothing$ then $\tau \equiv \langle\langle\pi_1\circ\pi_1, \pi_2\rangle, \pi_2\circ\pi_1\rangle : (C\times A)\times B\to(C\times B)\times A$. Show that if $\vdash\Gamma,x{:}\alpha, y{:}\beta, \Gamma' \supset M{:}\gamma$ then $\vdash\Gamma,y{:}\beta,x{:}\alpha, \Gamma' \supset M{:}\gamma$ (cf. SEQvsNAT) and

$$[\![\Gamma,x{:}\alpha, y{:}\beta, \Gamma' \supset M{:}\gamma]\!] = [\![\Gamma,y{:}\beta,x{:}\alpha, \Gamma' \supset M{:}\gamma]\!] \circ \tau$$

The next step in understanding the interpretation is to prove the following, by induction on the length of derivations.

Lemma *(substitution)*

If $\vdash\Gamma,x{:}\alpha \supset M{:}\beta$, and $\vdash\Gamma \supset N{:}\alpha$ then

(1) $\vdash\Gamma \supset [N/x]M{:}\beta$

(2) $[\![\Gamma \supset [N/x]M{:}\beta]\!] = [\![\Gamma,x{:}\alpha \supset M{:}\beta]\!]\circ\langle id, [\![\Gamma \supset N{:}\alpha]\!]\rangle$

Proof

(1) By induction on the proof 's length of $\Gamma,x{:}\alpha \supset M{:}\beta$. The interesting case arises when the last deduction is an $(\to I)$: $\Gamma,x{:}\alpha,y{:}\beta \supset M{:}\beta' \Rightarrow \Gamma,x{:}\alpha \supset \lambda y{:}\beta.M{:}\beta\to\beta'$. Observe: $[N/x](\lambda y{:}\beta.M) \equiv \lambda y{:}\beta.[N/x]M$. Also we can apply the inductive hypothesis on $\Gamma,y{:}\beta,x{:}\alpha \supset M{:}\beta'$ (note the exchange on the assumptions) to get $\Gamma,y{:}\beta\supset[N/x]M{:}\beta'$ from which $\vdash\Gamma\supset[N/x](\lambda y{:}\beta.M){:}\beta\to\beta'$ follows by $(\to I)$.

(2) We will use the exercises above on the interpretation of weakening and exchange. Again we proceed by induction on the proof's length of $\Gamma,x{:}\alpha \supset M{:}\beta$ and we just consider the case for the $(\to I)$. Let

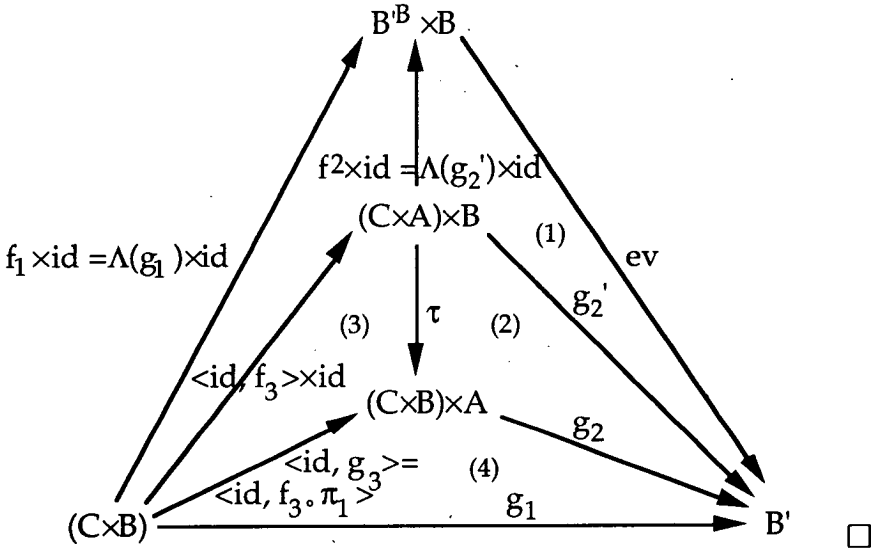
$f_1 = [\![\Gamma \supset \lambda y{:}\beta.[N/x]M{:}\beta\to\beta']\!] : C\to B'^B$
$f_2 = [\![\Gamma,x{:}\alpha \supset \lambda y{:}\beta.M{:}\beta\to\beta']\!] : C\times A\to B'^B$
$f_3 = [\![\Gamma \supset N{:}\alpha]\!] : C\to A$

$g_1 = [\![\Gamma,y{:}\beta \supset [N/x]M{:}\beta']\!] : C\times B\to B'$
$g_2 = [\![\Gamma,y{:}\beta,x{:}\alpha \supset M{:}\beta']\!] : (C\times B)\times A\to B'$
$g_3 = [\![\Gamma,y{:}\beta \supset N{:}\alpha]\!] : C\times B\to A$
$g_2' = [\![\Gamma,x{:}\alpha,y{:}\beta\supset M{:}\beta']\!] : (C\times A)\times B\to B'$

[4] DeBrujin conventions for the representation of variables as distances from the respctive binders, as well as standard implementations of environments in abstract machines follow related ideas.

We have: (1) $\Lambda(g_2') = f_2$, $ev\circ(\Lambda(g_2')\times id) = ev\circ(f_2\times id) = g_2'$.
(2) $g_2' = g_2\circ\tau$
(3) $g_3 = f_3\circ\pi_1$, $\langle id, g_3\rangle = \langle id, f_3\circ\pi_1\rangle$, $\langle id, f_3\rangle\times id\circ\tau = \langle id, g_3\rangle$
(4) $g_1 = g_2\circ\langle id, g_3\rangle$

Also: $f_1 = \Lambda(g_1)$, $f_1\times id = \Lambda(g_1)\times id$. We have to show $f_1 = f_2\circ\langle id, f_3\rangle$. Since $f_1 = \Lambda(g_1)$ it is enough to show: $ev\circ f_2\times id\circ\langle id, f_3\rangle\times id = g_1$ that follows by $ev\circ f_2\times id = g_2' = g_2\circ\tau$, and $\langle id, f_3\rangle\times id = \tau^{-1}\circ\langle id, g_3\rangle$. Here is the diagram:



□

4. From CCCs to Lambda Theories

In this section we analyse the problem of determining which kind of equivalence is induced by the interpretation of the simply typed lambda calculus in a CCC. It turns out that the resulting equivalence is closed under βη-conversion, and under certain natural rules that make the equivalence into a congruence.

In the first place we describe a formal system to reason about judgments of the shape $\Gamma\supset M=N:\alpha$. Such judgments should be read as saying that the typed terms M and N are equal in the context Γ and the type α.

(α) $\Gamma\supset \lambda x{:}\alpha.N: \alpha\to\beta$, y not in $\Gamma \Rightarrow \Gamma\supset \lambda y{:}\alpha.[y/x]N = \lambda x{:}\alpha.N: \alpha\to\beta$
(β) $\Gamma\supset (\lambda x{:}\alpha.M)N:\beta \Rightarrow \Gamma\supset (\lambda x{:}\alpha.M)N = [N/x]M: \beta$
(η) $\Gamma\supset \lambda x{:}\alpha.Mx: \alpha\to\beta$, x not in $M \Rightarrow \Gamma\supset \lambda x{:}\alpha.Mx=M: \alpha\to\beta$.
(weak) $\Gamma\supset M=N:\alpha$, x not in $\Gamma \Rightarrow \Gamma, x{:}\beta\supset M=N:\alpha$
(refl) $\Gamma\supset M:\alpha \Rightarrow \Gamma\supset M=M:\alpha$
(sym) $\Gamma\supset M=N:\alpha \Rightarrow \Gamma\supset N=M:\alpha$
(trans) $\Gamma\supset M=N:\alpha$, $\Gamma\supset N=P:\alpha \Rightarrow \Gamma\supset M=P:\alpha$
(ξ) $\Gamma, x{:}\alpha\supset M=N:\beta \Rightarrow \Gamma\supset \lambda x{:}\alpha.M= \lambda x{:}\alpha.N:\alpha\to\beta$
(apl) $\Gamma\supset M= N:\alpha\to\beta$, $\Gamma\supset M'= N':\alpha \Rightarrow \Gamma\supset MM' = NN':\beta$

Exercise: Show by induction on the length of the derivation that if $\vdash\Gamma\supset M=N:\alpha$ then $\vdash\Gamma\supset M:\alpha$ and $\vdash\Gamma\supset N:\alpha$

The system just introduced gives the *pure* $\lambda\beta\eta$-theory of term equivalence. If we want to add *proper* axioms then we arrive at the following

Definition *(λ-theory)*

A lambda theory T is a collection of judgments of the shape $\Gamma\supset M=N:\alpha$ such that: (i) $\Gamma\supset M=N:\alpha \in T$ implies $\vdash\Gamma\supset M:\alpha$ and $\vdash\Gamma\supset N:\alpha$, and (ii) T is closed under the rules of the *pure* $\lambda\beta\eta$-theory.

Theorem *(every CCC generates a λ-theory)*

Let **C** be a CCC and let $[\![\ ]\!]$ be some standard interpretation of the simply typed lambda calculus defined over **C**. Then the collection:

$$Th(\mathbf{C}) \triangleq \{\Gamma\supset M=N:\alpha \mid \vdash\Gamma\supset M:\alpha, \vdash\Gamma\supset N:\alpha, [\![\Gamma\supset M:\alpha]\!] = [\![\Gamma\supset N:\alpha]\!]\}$$

is a lambda theory.

Proof

We have to check that Th(**C**) is closed under the rules presented above.

(α) Observe that $[\![\ ]\!]$ is invariant w.r.t. the names of bound variables.

(β) Observe: $[\![\Gamma\supset (\lambda x{:}\alpha.M)N:\beta]\!] = ev\circ\langle[\![\Gamma\supset\lambda x{:}\alpha.M:\alpha\to\beta]\!], [\![\Gamma\supset N:\alpha]\!]\rangle =$

$ev\circ\langle\Lambda(f), g\rangle$, where $f = [\![\Gamma, x{:}\alpha\supset M:\beta]\!]$, $g = [\![\Gamma\supset N:\alpha]\!]$.

Also: $[\![\Gamma\supset[N/x]M:\ \beta]\!] = f\circ\langle id, g\rangle$, by the *substitution lemma*,

$f\circ\langle id, g\rangle = ev\circ(\Lambda(f)\times id)\circ\langle id, g\rangle$, by def. CCC.

And: $\langle\Lambda(f), g\rangle = (\Lambda(f)\times id)\circ\langle id, g\rangle$.

(η) Observe: $[\![\Gamma\supset \lambda x{:}\alpha.Mx:\ \alpha\to\beta]\!] = \Lambda([\![\Gamma, x{:}\alpha\supset Mx:\beta]\!]) =$

$\Lambda(ev\circ\langle[\![\Gamma, x{:}\alpha\supset M:\alpha\to\beta]\!], [\![\Gamma, x{:}\alpha\supset x:\alpha]\!]\rangle) =$

$\Lambda(ev\circ\langle[\![\Gamma\supset M:\alpha\to\beta]\!]\circ\pi_1, \pi_2\rangle) = \Lambda(ev\circ([\![\Gamma\supset M:\alpha\to\beta]\!]\times id)) =$

$[\![\Gamma\supset M:\alpha\to\beta]\!]$.

(weak) Use exercise (IntWeak).

(refl), (sym), (trans) Clearly Th(**C**) is an equivalence.

(ξ) (apl) follow by the definition of the interpretation of abstraction and application. □

Note: There is a minimal $\lambda\beta\eta$-theory, and a maximal one. Between these two extremes there are infinitely many λ-theories. It is easy to observe this fact by interpreting the basic types as finite sets and by considering the resulting lambda-theory. Hint: consider the $\lambda\beta\eta$ normal forms of type $(t\to t)\to(t\to t)$.

5. From Lambda Theories to CCCs.

In this section we show how to generate a CCC starting from a lambda theory. The construction consists essentially in taking types as objects of the category and (open) terms quotiented by the lambda theory as morphisms. We follow the following steps:

(1) We extend the language with constructors for terminal object and product, as well as the relative equations:

Types: $\alpha ::= t \mid s... \mid (\alpha \wedge \alpha) \mid (\alpha \to \alpha)$

Terms: $M ::= x \mid y... \mid * \mid \langle M, M \rangle \mid (\pi_1\ M) \mid (\pi_2\ M) \mid (\lambda x{:}\alpha.M) \mid (MM)$

Typing Rules: rules of the simply typed calculus, plus rules for conjunction plus

(*) $\Rightarrow \Gamma \supset *: 1$

Equations: rules of the pure $\lambda\beta\eta$-theory plus

(*) $\Gamma \supset M: 1 \Rightarrow \Gamma \supset M = *$

(π_1) $\Gamma \supset \langle M, N \rangle: \alpha \wedge \beta \Rightarrow \Gamma \supset \pi_1 \langle M, N \rangle = M: \alpha$

($\pi 2$) $\Gamma \supset \langle M, N \rangle: \alpha \wedge \beta \Rightarrow \Gamma \supset \pi_2 \langle M, N \rangle = N: \beta$

(SP) $\Gamma \supset M: \alpha \wedge \beta \Rightarrow \Gamma \supset \langle \pi 1 M, \pi 2 M \rangle = M: \alpha \wedge \beta$ □

We denote by T' the collection of judgment provable in the λ-theory T extended with the constructors and rules given above.

(2) We now associate to T' a CCC C(T') as follows.

- The objects are the types of the extended language.
- The morphisms are equivalences classes of terms according to the equivalence induced by T'. More precisely:

$$C(T')[\alpha, \beta] = \{ [x{:}\alpha \supset M: \beta] \mid \vdash x{:}\alpha \supset M{:}\beta \}$$

where: $[x{:}\alpha \supset M: \beta] = \{y{:}\alpha \supset N: \beta \mid \varnothing \supset \lambda x{:}\alpha.M = \lambda y{:}\alpha.N: \alpha \to \beta \in T'\}$.

- The special maps associated to every CCC are defined as follows:

Identity: $[x{:}\alpha \supset x{:}\alpha]$.

Composition: $[x{:}\beta \supset M: \gamma] \circ [y{:}\alpha \supset N: \beta] = [y{:}\alpha \supset [N/x]M: \gamma]$

Terminal Object: $*_\alpha = [x{:}\ \alpha \supset *{:}1]$

Projections: $\pi\alpha = [x{:}\ \alpha \wedge \beta \supset \pi_1\ x{:}\ \alpha]$, $\pi\beta = [x{:}\ \alpha \wedge \beta \supset \pi_2\ x{:}\ \beta]$

Pairing: $\langle [x{:}\gamma \supset M{:}\ \alpha], [x{:}\gamma \supset N{:}\ \beta] \rangle = [x{:}\gamma \supset \langle M, N \rangle{:}\ \alpha \wedge \beta]$

Evaluation: $ev_{\alpha,\beta} = [x{:}\ (\alpha \to \beta) \wedge \alpha \supset (\pi_1\ x)(\pi_2\ x){:}\ \beta]$

Uncurrying: $\Lambda([x{:}\ \gamma \wedge \alpha \supset M{:}\ \beta]) = [y{:}\ \gamma \supset \lambda z{:}\alpha.[\langle z,y \rangle / x]M{:}\ \alpha \to \beta]$

- We leave to the reader the verification of the equations associated to a CCC.

(3) Finally we have to verify that the lambda theory associated to C(T') is exactly T'. Idea: verify $[\![x_1{:}\alpha_1,...,x_n{:}\alpha_n \supset M{:}\alpha]\!] = [\ x{:}\gamma \supset [\pi_{n,i}\ x/x]M{:}\alpha\ ]$, where $\gamma \equiv (..(1 \times \alpha_1)...\times \alpha_n)$. □

We can summarize our constructions in the following

Theorem *(from lambda theories to CCC)*

Given any lambda theory T over the simply typed calculus with products and terminal object we can build a CCC C(T) such that the lambda theory associated to C(T) coincides with T.

Notes

(1) We refer the reader to Lambek&Scott[86] for an abstract categorical framework in which the constructions described here can be presented. We mention in particular that it is possible to see such constructions as representing an equivalence between a "category of CCCs" and a "category of lambda theories".

(2) It is possible to strengthen the previous theorem by considering a theory T over the simply typed lambda calculus *without* products and terminal object. Then one needs to show that it is possible to add conservatively to T the equations (*), (π_1), (π_2), and (SP) (see Curien[86], chpt. 1, for a description of suitable proof techniques). □

6. Type Free Lambda Calculus and its Interpretation in CCCs

The type-free λβ calculus is simply described as follows:

Terms: $M ::= x \mid y... \mid (\lambda x.M) \mid (MM)$

β-reduction: $(\lambda x.M)N \rightarrow [N/x]M$

In order to use the work done in the typed case it is useful to represent the type free calculus as a typed calculus with a special type δ, and the following typing rules:

(asmp) $x: \delta \in \Gamma \Rightarrow \Gamma \supset x: \delta$

$(\rightarrow I)$ $\Gamma, x:\delta \supset M: \delta \Rightarrow \Gamma \supset \lambda x:\delta.M: \delta$

$(\rightarrow E)$ $\Gamma \supset M: \delta \ , \ \Gamma \supset N: \delta \Rightarrow \Gamma \supset MN: \delta$

Observe that if a type $\delta \rightarrow \delta$ could magically contract into a type δ in the clause $(\rightarrow I)$, and vice versa, if the type δ could magically expand into a type $\delta \rightarrow \delta$ in the clause $(\rightarrow E)$ then we would have the same rules as in the simply typed lambda calculus. In other words we can apply the standard apparatus provided we have a type whose elements can be seen both as arguments and as functions.

We make this intuition formal by interpreting δ as an object D in a CCC **C** such that its functional space D^D can be *retracted* into D. Namely there is a (retraction) pair (i, j) such that: $i: D^D \rightarrow D$, $j: D \rightarrow D^D$, $j \circ i = id$. We call such object D *reflexive* and write sometimes $D^D \triangleleft D$.

Example *(reflexive objects in Dcpo)*

We describe a set-theoretic construction that produces reflexive objects in **Dcpo**. Such objects are usually called *graph-models*. Let A be a non-empty set equipped with an injective coding $< , >: P_{fin}(A)\times A \rightarrowtail A$ ($P_{fin}(A)$ is the collection of finite subsets). In the following denote with a, b,... elements of A; with α, β,... elements of $P_{fin}(A)$; and with X, Y,... elements of the powerset P(A) with the order given by the containment. Define for $f \in$ **Dcpo**[P(A), P(A)]

$$Graph(f) \triangleq \{<\alpha , a> \mid a \in f(\alpha)\}$$

Vice versa for $X \in P(A)$, define:

$$Fun(X)(Y) \triangleq \{a \mid \exists \alpha.(<\alpha , a> \in X, \alpha \subseteq Y)\}$$

Proposition

Given any non-empty set A with an injective coding $< , >: P_{fin}(A)\times A \rightarrowtail A$ the complete lattice P(A) is a reflexive object in **Dcpo**, via the morphisms Graph and Fun.

Proof

On P(A) the order is given by the containment relation. On the exponent **Dcpo**[P(A), P(A)] the order is pointwise. The following assertions follow directly from the definitions: (i) Graph is monotone: $f \leq g \Rightarrow Graph(f) \subseteq Graph(g)$. (ii) Graph preserves directed sets: $\{f_i\}_{i\in I}$ directed $\Rightarrow Graph(\sqcup_{i\in I}\{f_i\}) \subseteq \cup_{i\in I} Graph(f_i)$, as: $<\alpha,a> \in Graph(\sqcup_{i\in I}\{f_i\}) \Leftrightarrow a \in (\sqcup_{i\in I}\{f_i\})(\alpha) = \cup_{i\in I} f_i(\alpha) \Rightarrow a \in f_i(\alpha)$, for some $i \in I$. (iii) Fun is monotone in both arguments: $X \subseteq X'$, $Y \subseteq Y' \Rightarrow Fun(X)(Y) \subseteq Fun(X')(Y')$. (iv) Fun is continuous in both arguments: $\{X_i\}_{i\in I}$, $\{Y_j\}_{j\in J}$ directed $\Rightarrow$ $Fun(\cup_{i\in I}X_i)(\cup_{j\in J}Y_j) \subseteq \cup_{i\in I, j\in J} Fun(X_i)(Y_j)$. (v) (Graph, Fun) is a retraction: $Fun(Graph(f))(X) = f(X)$. Notice that this is the only condition that depends on the assumption that the coding is injective. □

This construction is parametric w.r.t. the choice of the set A and of the coding $< , >$. If one defines a coding $< , >: P_{fin}(\omega)\times\omega \rightarrowtail \omega$, where ω is the collection of natural numbers, then one obtains a family of reflexive objects also known as *$P\omega$ graph models*.

Exercise: Build an example of a coding $< , >: P_{fin}(\omega)\times\omega \rightarrowtail \omega$.

If we define the coding out of the following "free construction", we get what is known as the *D_A graph models*. Let At be a non empty set whose elements are not pairs. Define:

$$A_0 \triangleq At$$
$$A_{n+1} \triangleq A_n \cup \{(\alpha, a) \mid \alpha \in P_{fin}(A_n), a \in A_n\}$$
$$A \triangleq \cup_{n<\omega} A_n$$

Then verify that $< , >: P_{fin}(A)\times A \rightarrow A$ defined as $<\alpha, a> \triangleq (\alpha, a)$ is the desired coding.

Having verified the existence of interesting CCCs with reflexive objects we can now introduce the interpretation of the type free λβ calculus in such structures:

(asmp) $[\![(x_1: \delta),\ldots, (x_n: \delta) \supset x_i: \delta]\!] = \pi_{n,i}$

(→I) $[\![\Gamma \supset \lambda x{:}\delta.M{:}\delta]\!] = i \circ \Lambda([\![\Gamma, x{:}\delta \supset M{:}\delta]\!])$

(→E) $[\![\Gamma \supset MN{:}\delta]\!] = ev \circ \langle j \circ [\![\Gamma \supset M{:}\delta]\!], [\![\Gamma \supset N{:}\delta]\!] \rangle$

This is the same as the interpretation of the simply typed calculus with the remarkable exception of the insertion of the maps i, j which collapse the hierarchy of types to D. As for the theory of equality induced we have the following

Definition *(type-free λβ-theory)*

A (type-free) λβ-theory T is a collection of judgments of the shape $\Gamma \supset M=N:\delta$ such that: (i) $\Gamma \supset M=N:\delta \in T$ implies $\vdash \Gamma \supset M:\delta$ and $\vdash \Gamma \supset N:\delta$, and (ii) T is closed under the following rules:

(α) $\Gamma \supset \lambda x{:}\delta.N{:}\ \delta$, y not in $\Gamma \Rightarrow \Gamma \supset \lambda y{:}\delta.[y/x]N = \lambda x{:}\delta.N{:}\ \delta$

(β) $\Gamma \supset (\lambda x{:}\delta.M)N{:}\delta \Rightarrow \Gamma \supset (\lambda x{:}\delta.M)N = [N/x]M{:}\ \delta$

(weak) $\Gamma \supset M=N:\delta$, x not in $\Gamma \Rightarrow \Gamma, x{:}\delta \supset M=N:\delta$

(refl) $\Gamma \supset M:\delta \Rightarrow \Gamma \supset M=M:\delta$

(sym) $\Gamma \supset M=N:\delta \Rightarrow \Gamma \supset N=M:\delta$

(trans) $\Gamma \supset M=N:\delta$, $\Gamma \supset N=P:\delta \Rightarrow \Gamma \supset M=P:\delta$

(ξ) $\Gamma, x{:}\delta \supset M=N:\delta \Rightarrow \Gamma \supset \lambda x{:}\delta.M= \lambda x{:}\delta.N{:}\delta$

(apl) $\Gamma \supset M= N:\delta$, $\Gamma \supset M'= N':\delta \Rightarrow \Gamma \supset MM' = NN':\delta$

Theorem *(every CCC with a reflexive object generates a λ-theory)*

Let **C** be a CCC with a reflexive object and [\[\]] some standard interpretation of the type-free lambda calculus defined over **C**. Then the collection:

$Th(\mathbf{C}) \triangleq \{\Gamma \supset M=N:\delta \mid \vdash \Gamma \supset M:\delta, \vdash \Gamma \supset N:\delta, [\![\Gamma \supset M:\delta]\!] = [\![\Gamma \supset N:\delta]\!] \}$

is a (type-free) lambda theory.

Proof *(hint)*

The proof of this result follows the same schema as in the typed case. The crucial point is to verify the substitution lemma. □

Appendix 1: CCC of Retractions

In this section we describe a general construction, called "Karoubi envelope" that, given a reflexive object D in a CCC, produces a new CCC of "retractions over D". Apart for its intrinsic interest this construction can be used to reverse the previous theorem, namely to show that every type-free λ-theory is the theory induced by a reflexive object in a CCC; a result very much in the spirit of the one for the typed case in section 5. Given a λ-theory, the idea is to build the category of retractions over the monoid of terms and composition and show that this forms a

CCC with a reflexive object (see Scott[80], for details).

Definition *(Karoubi envelope)*

Let **C** be a CCC and D a reflexive object in **C**. The Karoubi envelope is the category **Ret(D)** whose *objects* are retractions: $Ret(D) \triangleq \{r: D \to D \mid r \circ r = r\}$ and whose *morphisms* from r to s are $Ret[r, s] \triangleq \{f: D \to D \mid s \circ f \circ r = f\}$.

Definition*(CCC with fix-points)*

A CCC **C** has fix-points if for every object A there is a morphism $Fix_A: A^A \to A$ such that, for any morphism $f: A \to A$, $ev \circ \langle Fix_A, f\rangle = Fix_A$.

Exercise: Let **Cpo** be the full subcategory of **Dcpo** whose objects have a least element. Show that this is a cartesian closed category with fixpoints. Hint: consider $Fix_D = \lambda f: D \to D.\ f^n(\perp_D)$, where $\perp_D$ is D least element (cf. chapter 1).

Theorem *(Ret(D) as a CCC)*

If **C** is a CCC and D is a reflexive object in **C** then Ret(D) is a CCC with fixpoints.

Proof

The whole proof is a matter of translating λ-calculus coding into the categorical language and check that everything goes through. Here we just remind the reader of the encoding. When we write λ-terms for building morphisms it is intended that one has to take the *interpretations* of such lambda terms.

In the type free λβ-calculus we can define a fixed point combinator: $Y \triangleq \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$, and terms for pairing and projections: $[\ ,\ ] \triangleq \lambda x.\lambda y.\lambda p.pxy$, $p_1 \triangleq \lambda p.p(\lambda x.\lambda y.x)$, $p_2 \triangleq \lambda p.p(\lambda x.\lambda y.y)$ so that $p_1[x, y]=x$, $p_2[x, y]=y$.

Terminal Object. In a CCC we have a unique morphism $!_D: D \to 1$, also if we take $Y(\lambda x.x)$ we get a morphism from 1 to D. From this follows $1 \lhd D$. Then take the retraction determined by 1 as the terminal object in Ret(D).

Product. The pairing and projections defined above show that $D \times D \lhd D$ via a retraction that we denote with $\langle\ ,\ \rangle: D \times D \to D$, $(\pi_1, \pi_2): D \to D \times D$. If r and s are retractions then define their product as: $r \times s \triangleq \lambda x.\langle r(\pi_1(x)), s(\pi_2(x))\rangle$.

Exponent. If r and s are retractions then define their exponent as: $r \to s \triangleq \lambda x.i(s \circ j(x) \circ r)$. □

Appendix 2: Embedding Algebras in Lambda Calculus

In this section we use Ret(D) as a frame for an abstract formulation of Engeler's theorem (Engeler[81]) on the embedding of algebras in "λ-models".

Let **C** be a CCC and let D be a reflexive object in **C**. Let $\Sigma \equiv \{\sigma_i^{n_i}\}_{i \in In}$ be a *finite signature*, that is a finite collection of names of operators f_i with the relative arity n_i. We are interested in a notion of *Σ-algebra* in which the carriers are objects in Ret(D) and the operators are maps in Ret(D) of the appropriate type.

Definition *(Σ_D-algebra)*

Let $\Sigma=\{\sigma_i^{n_i}\}_{i\in I_n}$ be a finite signature, **C** be a CCC, and let D be a reflexive object in **C**. A Σ_D-algebra is a pair $(r, \{f_i^{n_i}\}_{i\in I_n})$ where $r\in Ret(D)$ and $f_i^{n_i}$: $r\to r$ for $i\in I_n$ are morphisms of the appropriate type in Ret(D).

By morphism of Σ_D-algebras h: $(r, \{f_i^{n_i}\}_{i\in I_n})\to(r', \{g_i^{n_i}\}_{i\in I_n})$ we intend a morphism h: $r\to r'$ in Ret(D) that preserves the operations. In the following proof we make an assumption on the "concreteness" of the category **C**. Technically this boils down to ask that the terminal object, 1, is a *generator*, or, equivalently, that **C** has *enough points*, i.e. given morphisms f, g: $A\to B$ one has:
$\forall x: 1\to A.\ f\circ x = g\circ x \Rightarrow f = g$.

Theorem *(Embedding Σ_D-algebras)*

Let $\Sigma=\{\sigma_i^{n_i}\}_{i\in I_n}$ be a finite signature, **C** be a CCC with enough points, and D be a reflexive object in **C**. Then there is a Σ_D-algebra $(id, \{F_i^{n_i}\}_{i\in I_n})$ that is 'weakly universal' in the category of Σ_D-algebras and monomorphisms. In other terms there is always a monomorphism from a Σ_D-algebra $(r, \{f_i^{n_i}\}_{i\in I_n})$ to $(id, \{F_i^{n_i}\}_{i\in I_n})$.

Proof

For the sake of simplicity we just consider the case $\Sigma=\{\sigma^2\}$. Assume that $<\ ,\ >$, and π_1, π_2 are respectively the pairing and the projections built out of the λ-calculus as usual. We take $F \triangleq \lambda x.\ \pi_2 x$. Given the Σ_D-algebra $(r, \{f\})$ define recursively a map ρ: $D\to D$ as follows:

$$\rho(a) = <a, \lambda x.\ \rho(fa(\pi_1 x))>$$

In the first place let us observe that ρ is a mono as (we use the enough points hyp.):

$$\rho(a) = \rho(b) \Rightarrow a = \pi_1(\rho(a)) = \pi_1(\rho(b)) = b.$$

Clearly $\rho\circ r$: $r\to id$ in Ret(D) as $\rho\circ r = id\circ\rho\circ r\circ r$. Also since $r(f(rx)(ry)) = f(rx)(ry)$ we have:

$F\rho(ra)\rho(rb) = (\lambda x.\ \rho(f(ra)(\pi_1 x)))\rho(rb) = \rho(f(ra)(\pi_1\rho(rb))) = \rho(f(ra)(rb)) = (\rho\circ r)(f(ra)(rb))$.

Therefore $\rho\circ r$: $(r, \{f\})\to(id, \{F\})$ is a Σ_D-algebras mono-morphism. □

Notes

(1) One may say that the moral of this result is that by means of a recursion trick it is possible to put into the data the information on the behaviour of the operations defined on them.

(2) The following describes a schema for coding finite signatures which suggests how to generalize the previous proof. Given $\Sigma = \{\sigma, f_0^0, ..., f_n^n\}$ define:

$$\begin{aligned}\rho(a) = <a,\ & <\rho(f_1 a),\\ & <\lambda x.\ \rho(f_2\ a\ (\pi_1 x)),\\ & \quad ....\\ & <\lambda x_1 ...\ \lambda x_{n-1}.\ \rho(f_n\ a\ (\pi_1 x_1)...(\pi_1 x_n), *>....>\end{aligned}$$

where:

$F_1 = \lambda x.\ \pi_1(\pi_2(\pi_2 x))$
$F_2 = \lambda x.\ \pi_1(\pi_2(\pi_2(\pi_2 x)))$
...
$F_n = \lambda x.\ \pi_1(\pi_2(...(\pi_2 x)..)$, with n π_2 . □

3. Cartesian Closed Categories of Algebraic Dcpos

Contents: 1. Continuous dcpos, 2. Cartesian Closed Categories of Algebraic (d)cpos, 3. The Two Maximal Cartesian Closed Full Subcategories of Acpo, 4. The Four Maximal Cartesian Closed Full Subcategories of Adcpo.

This chapter is more mathematical in spirit. We shall provide a finer analysis of algebraicity, towards the aim of finding the maximal full subcategories of **Acpo** and **Adcpo** which are cartesian closed. The basic result, which was conjectured in Plotkin[83], and first proved in Smyth[83], is that there exists a maximum cartesian closed full subcategory of **ω-Acpo** (the category of ω-algebraic cpos). A. Jung, sharply analyzing this result, has provided complete answers for **Acpo**, **Adcpo** (and **ω-Adcpo**) as well. We shall follow Jung[88] here.

One first needs some background on a class of dcpos where approximations exist without being necessarily compact. This class of *continuous* dcpos was studied in depth from a mathematical perspective quite independently from the interest of dcpos in computer science (we refer to Gierz&al.[80]). It contains the class of algebraic dcpos. The interest of this class to us is that retracts of algebraic dcpos (definition below) are not algebraic in general, but are continuous. Much of the technical work involved in our quest for maximal cartesian closed subcategories of (d)cpos involves retracts, on which it is easier to define crucial functions than on the full space, just because these retracts are smaller!

We introduce continuous dcpos (section 1). We define two cartesian closed categories of algebraic dcpos, cpos respectively: the profinite dcpos (when they are cpos they are called bifinite) and the L-cpos (section 2). We show that the algebraic L-cpos and the bifinite domains yield the two maximal cartesian closed full subcategories of **Acpo**, and derive the result for **ω-Acpo** with little extra work (section 3). Finally, the situation with **Adcpo** is explored (section 4). There are more results in Jung[88].

Besides the beauty and importance of these results, the main purpose of this chapter is to introduce the reader to some involved mathematical exercise in domain theory.

1. Continuous dcpos

In order to define algebraic dcpos, we *first* introduced the notion of compact element, then we defined algebraicity. The definition of continuous dcpos is more direct.

Definition *(Continuous dcpo)*

Let D be a dcpo. For elements $x,y \in D$, we say that x is *way-below* y, and write $x \ll y$, if

$$\Delta \text{ directed}, \; y \leq \bigsqcup \Delta \Rightarrow \exists \delta \in \Delta \; x \leq \delta .$$

D is *continuous* if for any x in D, $\{y \mid y \ll x\}$ is directed and has x as lub.

Notice that by definition x is compact iff $x \ll x$. We leave as an exercise the proof of the following easy properties:

- if $x \ll y$, then $x \leq y$,
- if $x' \leq x \ll y \leq y'$, then $x' \ll y'$.

Clearly, algebraic dcpos are continuous, but not conversely:

Exercise CONT-NONALG: Show that the interval [0,1] of real numbers is continuous but not algebraic.

The following are easy and useful observations, which we prove in detail to practice with the new definition.

Lemma =WAY-BELOW

In a continuous dcpo, $x \ll y$ holds iff Δ directed , $y = \bigsqcup \Delta \Rightarrow \exists \delta \in \Delta \; x \leq \delta$.

Proof

We write $x \ll' y$ when $y = \bigsqcup \Delta \Rightarrow \exists \delta \in \Delta \; x \leq \delta$, for all directed Δ. Suppose $x \ll' y$. Since $y = \bigsqcup \{y' \mid y' \ll y\}$, we have $x \leq y'$ for some y'. Hence $x \ll y$ since $x \leq y' \ll y$. □

Lemma $\subseteq$WAY-BELOW

Let D be a dcpo and $x \in D$. If $A \subseteq \{y \mid y \ll x\}$ is directed and $x = \bigsqcup A$, then $\{y \mid y \ll x\}$ is directed and $x = \bigsqcup \{y \mid y \ll x\}$.

Proof

If $y \ll x$, $y' \ll x$, by definition $y \leq a$, $y' \leq a'$ for some $a,a' \in A$. We have by directedness $a,a' \leq y''$ for some $y'' \in A$. Hence $y,y' \leq y'' \in \{y \mid y \ll x\}$. The inequality $x \leq \bigsqcup \{y \mid y \ll x\}$ follows from the obvious inequality $\bigsqcup A \leq \bigsqcup \{y \mid y \ll x\}$. □

The following is a more surprising and crucial property.

Lemma *(Interpolation)*

In a continuous dcpo D, if $x \ll y$, then there exists $z \in D$ s.t. $x \ll z \ll y$.

Proof

Consider $A \triangleq \{a \in D \mid \exists a' \in D \; a \ll a' \ll y\}$. If we show that A is directed and $\bigsqcup A = y$, then we can conclude, since by definition $x \ll y$ implies $x \leq a$ for some $a \in A$, hence $x \in A$. A is non-empty, since the directedness of $\{y' \mid y' \ll y\}$ implies its non emptiness. Thus one can find at least an $a' \ll y$, and then at least an $a \ll a'$. Suppose that $a \ll a' \ll y$ and $b \ll b' \ll y$. By directedness, there exists $c' \in D$ s.t. $a',b' \leq c' \ll y$. Hence

$a,b \ll c'$, and by directedness again $a,b \leq c \ll c'$ for some c, which is in A since $c \ll c' \ll y$. Hence A is directed. Since $a \ll a' \ll y$ implies $a \ll y$, one has clearly $\sqcup A \leq \sqcup\{y' \mid y' \ll y\}$. Conversely, if $y' \ll y$, then $\{y'' \mid y'' \ll y'\} \subseteq A$, hence $\sqcup\{y' \mid y' \ll y\} = \sqcup\{y'' \mid y'' \ll y' \ll y$ for some $y'\} \leq \sqcup A$. □

We move on to our next, more routine property:

Lemma *(Mubs of compacts)*
In a continuous dcpo D, minimal upper bounds (mubs) of finite sets of compact elements are compact.

Proof
Let $A \subseteq D_0$ be finite, and $x \in MUB(A)$. We have $x = \sqcup\{y \mid y \ll x\}$. Let $a \in A$. Since $a \ll a \leq x$, we have $a \ll x$. By directedness there exists $x' \in \{y \mid y \ll x\} \cap UB(A)$. But $x' \leq x$, $x \in MUB(A)$ then imply $x' = x$, and $x' \ll x$ then means exactly that x is compact. □

We move on to retractions. Retractions are defined in any category. An arrow r is a *retraction* or an *idempotent* if $r \circ r = r$. In **Dcpo** a *retract* of a dcpo D is the range r(D) of some retraction over D, with the induced ordering. In **Dcpo** still (more generally in an ordered category, see chapter 5), if a retraction r is such that $r \leq Id$, we say that r is a *projection*. Very simple retractions are the following ones:

Lemma *(Simple retractions)*
Fix a dcpo D and $d \in D$. Then $\downarrow x$ is a retract of D. If x is compact, then $\uparrow x$ is a retract of D too.

Proof
Indeed $\downarrow d = r(D)$ where $r(x) = x$ if $x \leq d$ and $r(x) = d$ otherwise. In order to get confidence, let us check that r is continuous. If $\sqcup\Delta \leq d$, then $\forall \delta \in \Delta\ \delta \leq d$, hence $r(\sqcup\Delta) = \sqcup\Delta = \sqcup r(\Delta)$. If $\neg(\sqcup\Delta \leq d)$, then $\exists \delta \in \Delta\ \neg(\delta \leq d)$. We have then $r(\delta) = d$, which implies $\sqcup r(\Delta) = d = r(\sqcup\Delta)$.

If x is compact, we have $\uparrow x = s(D)$ where $s(x) = x$ if $x \geq d$ and $s(x) = d$ otherwise. If $d \leq \sqcup\Delta$, then $\exists \delta \in \Delta$. We set $\Delta' = \Delta \cap \uparrow\delta$, and have obviously $s(\sqcup\Delta') = \sqcup s(\Delta')$, whence we deduce $s(\sqcup\Delta') = \sqcup s(\Delta')$. If $\neg(d \leq \sqcup\Delta)$, then $\forall \delta \in \Delta\ \neg(d \leq \delta)$, so that $s(\sqcup\Delta) = d = s(\Delta)$, hence $s(\sqcup\Delta) = \sqcup s(\Delta)$. □

Retractions are at the heart of our interest in continuous dcpos. Indeed, retracts of algebraic dcpos are not algebraic in general, but only continuous (see Exercise RETR-ALG-CONT).

Proposition *(Retracts of (continuous) dcpos)*
A retract r(D) of a dcpo D is a subdcpo. If D is continuous, then r(D) is continuous.

Proof

Let $\Delta \subseteq r(D)$ be directed. Then $r(\bigsqcup\Delta)=\bigsqcup r(\Delta)=\bigsqcup\Delta$, since $\forall \delta\in\Delta\ r(\delta)=\delta$. Suppose that $x\ll y\in r(D)$. We show $r(x)$ is way-below y in $r(D)$. If $y\leq\bigsqcup\Delta$, with $\Delta\subseteq r(D)$, then $x\ll y$ implies $x\leq\delta$ for some $\delta\in\Delta$; hence $r(x)\leq r(\delta)=\delta$. Since $y=r(y)=r(\bigsqcup\{x \mid x\ll y\})=\bigsqcup\{r(x) \mid x\ll y\}$, we can conclude by Lemma $\subseteq$WAY-BELOW. □

Exercise →RETRACT: This exercise shows a fundamental way of building retractions out of other retractions. Let D and E be dcpos and $r:D\to D$, $s:E\to E$ be retractions. Then show that $f \mapsto s\circ f\circ r$ defines a retraction on $D\to E$ whose image is isomorphic to $r(D)\to r(E)$.

Exercise RETR-ALG-CONT: This exercise shows that the continuous dcpos are exactly the retractions (actually the projections) of algebraic dcpos. Show that for any continuous dcpo D, D is a projection of Ide(D) (cf. Chapter 1, section 3).

We end the section with a simple but important property of projections: they are determined by their ranges.

Proposition *(Projections as ranges)*

For two projections p,p' over the same dcpo D, one has $p\leq p'$ iff $p(D)\subseteq p'(D)$.

Proof

If $y\in p(D)$, then $y=p(y)\leq p'(y)\leq y$ since $p\leq p'\leq id$, hence $y=p'(y)\in p'(D)$. Conversely, from $p(x)\in p'(D)$ we deduce $p(x)=p'(p(x))\leq p'(x)$. □

2. Cartesian Closed Categories of Algebraic (d)cpos

We introduce the profinite dcpos (a terminology due to C. Gunter) and show that they form a cartesian closed full subcategory of **Adcpo**. We recall that **Dcpo** is a cartesian closed category and that lubs of functions are defined pointwise. We call a projection (cf. section 1) *finite* when its range is finite.

Definition *(Profinite)*

A dcpo D is profinite if the finite projections form a directed set whose lub is the identity. We denote with **Prof** the category of profinite dcpos and continuous maps. A profinite cpo is called bifinite. We denote with **Bif** the category of bifinite cpos.

The terminology "bifinite" is due to P. Taylor, and comes from a more categorical characterization of profinite and bifinite dcpos to be found in chapter 5: they are limits and colimits at the same time (whence the "bi" in bifinite) of families of finite (d)cpos. The profinites enjoy the same property, so they might as well be called bifinite. The (ω)-bifinite domains have been first explored by Plotkin, under

the name of SFP (Sequence of Finite Projections).

Proposition (**Prof** *is a ccc)*

(1) Every profinite dcpo D is algebraic, with $D_o=\bigcup\{p(D) \mid p \text{ finite projection}\}$.

(2) Profinite dcpos (bifinite cpos, respectively) and continuous maps form a ccc.

Proof

(1) If D is profinite, then $x=\bigsqcup\{p(x) \mid p \text{ finite projection}\}$, for any x. By exercise ALG-$\exists\Delta$, it is enough to show that p(x) is compact, for any finite projection p. If $p(x)\leq\bigsqcup\Delta$, then $p(x)=p(p(x))\leq\bigsqcup p(\Delta)$. We now use finiteness of $p(\Delta)$: by fact FΔM, $p(x)\in p(\Delta)$. Take $\delta\in\Delta$ s.t. $p(x)= p(\delta)$. Then $p(x)\leq\delta$.

(2) It is enough to check that if D, E are profinite, then $D\times E$, $D\to E\in$ **Prof**. For $D\times E$, take the set of projections $(p\times q)$, where p, q are finite projections. For $D\to E$, define, for any pair of finite projections p, q on D, E:

$r(f) \triangleq x\mapsto q(f(p(x)))$.

One gets easily that f is the lub of these r's. We obtain that im(r) is finite from the following observations:

- there are finitely many functions from p(D) to q(E).
- every f s.t. r(f)=f restricts to a function $f:p(D)\to q(E)$, and is determined by this restriction: $f(x)=f(p(x))$ for any x. □

When there are only countably many finite projections, a profinite dcpo is called ω-profinite. This name is justified by the following exercise.

Exercise: Show that any ω-profinite dcpo is ω-algebraic.

We shall give an alternative characterization of profinite dcpos. We say that a partial order $(Y,\leq)$

- satisfies *property m* if for all finite subset X the set MUB(X) of mubs of X is complete, i.e. $\forall y\in Y\ \exists x\in MUB(X)\ x\leq y$,
- satisfies *property M* if it satisfies property m, with the additional condition that MUB(X) is finite for any finite subset X.

Theorem PROF-M∞

Let D be an ω-algebraic cpo. D is a bifinite domain iff the following hold:

- D_o satisfies property m,
- $U^*(X) \triangleq \bigcup(U^n(X) \mid n\in\omega\}$ is finite for any finite subset of compact elements X,where U is an operator on subsets defined by:

$U(X)=\bigcup\{MUB(Y) \mid Y\subseteq_{fin}X\}$.

Proof

Notice that in particular, for any X, $X\subseteq U(X)$, since $\{x\}\subseteq_{fin}X$ and $MUB(\{x\})=\{x\}$, and that if X is finite, $MUB(X)\subseteq U(X)$ since $X\subseteq_{fin}X$.

(⇒) We recall that $D_o = \bigcup\{p(D) \mid p \text{ finite projection}\}$. If $X \subseteq D_o$ is finite, then

$X \subseteq p(D)$ for some p by directedness and Proposition *(Projections as ranges)*. Call Z the set of mubs of X in p(D), which exists, is finite and complete (in p(D)), since p(D) is finite. We first show: $Z \subseteq MUB(X)$. Indeed suppose $z \in Z$, $z' \in UB(X)$ and $z'<z$. Then

- $p(z') \in UB(X)$, since $p(X)=X$,
- $p(z')<z$, since $p(z') \leq z'$.

This contradicts the definition of Z. Thus $Z \subseteq MUB(X)$. We next show that Z is a complete set of mubs of X in D. Take $y \in UB(X)$; then, as argued above, $p_n(y) \in UB(X)$, and by completeness of Z one may find $z \in Z$ s.t. $z \leq p(y)$, and a fortiori $z \leq y$. The completeness of Z forces $Z=MUB(X)$, which is therefore finite and complete. Moreover $Z \subseteq p(D)$. From there one deduces that $U^n(X) \subseteq p(D)$ for any n, observing that each subset of X is a fortiori included in p(D).

($\Leftarrow$) Let A be a finite set of compacts. Then

$\forall d \in D.\ U^*(A) \cap \downarrow d$ is directed.

This is shown as follows: if $x, x' \in U^*(A) \cap \downarrow d$, then $MUB(\{x,x'\}) \subseteq U^*(A)$ and by completeness $MUB(\{x,x'\}) \cap \downarrow d \neq \emptyset$. So we can set:

$p_A(d) \triangleq \sqcup(U^*(A) \cap \downarrow d)$.

It is left to the reader to check that this gives a directed set of finite projections having id as lub. □

Notice that, in the proof of ($\Leftarrow$) , we have used only mubs of pairs. The following result goes in the same direction.

Lemma PLOTKIN-2-PLOTKIN

Let $(D,\leq)$ be a partial order. If MUB(Y) is complete and finite for every subset Y s.t. $card(Y) \leq 2$, then MUB(X) is complete and finite for every finite subset X.

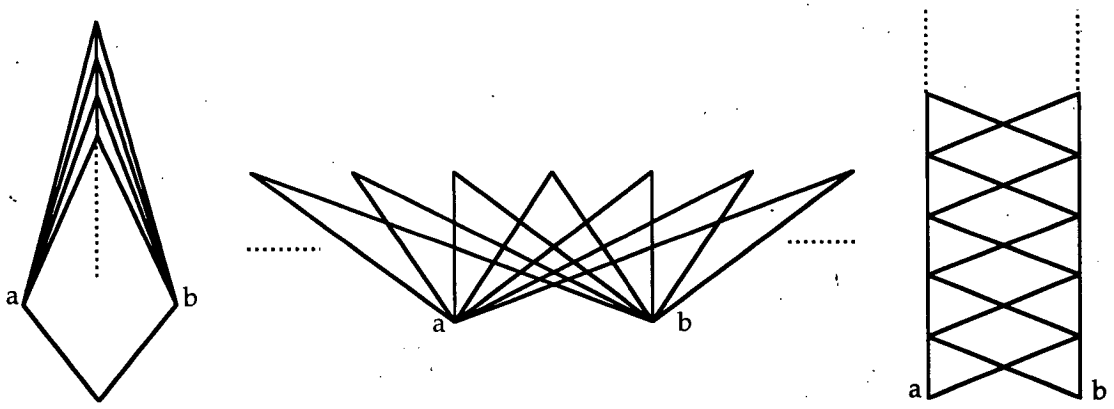
Proof

Let $X=\{a_1,\ldots,a_n\}$. We construct $M_2 \triangleq MUB(\{a_1,a_2\}),\ldots, M_n \triangleq \cup_{x \in M_{n-1}} MUB(\{x,a_n\})$. If x is an upper bound of X, then by completeness x dominates an element of M_2. Continuing in the same way we find an element y of M_n below x. Suppose now that moreover $x \in MUB(X)$; then x=y, since by construction $M_n \subseteq UB(X)$. We have proved $MUB(X) \subseteq M_n$. Since M_n is finite, MUB(X) is a fortiori finite. □

Exercises: (1) MUB-FC: Let X be finite. Show that if there exists $Y \subseteq_{fin} \uparrow X$ s.t. $\forall x \in \uparrow X$ $\exists y \in Y\ y \leq x$, then MUB(X) is finite and complete.
(2) Show that D is bifinite iff the conditions stated in Theorem PROF-M∞ hold, replacing the operator U by the operator $U'(X)=\cup\{MUB(Y) \mid Y \subseteq X \text{ and } card(Y) \leq 2\}$. Show that if D is bifinite, then $U^*(X)=U'^*(X)$.

The following picture illustrates the three essential ways in which an algebraic cpo

may fail to be bifinite. Example A was already studied in chapter 1 (section 5), where we proved that its function space is not algebraic. The same holds true of the function space of example C (the analysis of this example will lead to Proposition →ALG+M⇒BIF in section 3). The example (B) is less dramatic: its function space is algebraic, but not ω-algebraic. It is an example of L-cpo, which we shall introduce next.



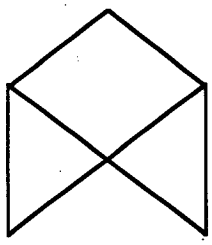
(A)¬U(X) complete (B) ¬U(X) finite (C) ¬U*(X) finite □

Definition L-PO

An L-partial order is a partial order D such that

$\forall A \subseteq_{fin} D\ \forall x \in UB(A)\ \exists!\ y\ \ y \leq x \in MUB(A)$.

An L-dcpo (L-cpo) is a L-partial order which is a dcpo (cpo). The following picture illustrates the minimal example of a finite partial order which is not an L-partial order.



Exercise 2L-DOM: Show that one can restrict Definition L-PO to the A's which have cardinal 2 without loss of generality (the uniqueness is essential).

Hence L-dcpos are "locally" complete: Any subset ↓y is a complete lattice. This is where the "L" comes from.

Exercise L-LATTICE: Show that a partial order D is a L-dcpo iff ↓y is a complete lattice for any $y \in D$.

The following proposition shows that L-cpos have something in common with

bifinite domains.

Proposition L-CPOS

A cpo D is an L-cpo iff D has property m and $U^*(A)=U(A)$ for all subsets A of D.

Proof

($\Rightarrow$) A fortiori finite subsets have a complete set of mubs. The second part of the statement reduces to showing $U^2(A)\subseteq U(A)$. Let $x\in MUB(B)$, for a finite $B\subseteq U(A)$, and let A_b be a finite subset of A of which b is a mub, for any $b\in B$. We show $x\in MUB(\cup\{A_b \mid b\in B\})$. By construction $x\in UB(\cup\{A_b \mid b\in B\})$. Suppose $x\geq y\in UB(\cup\{A_b \mid b\in B\})$. By property m, $y\geq b'$ for some mub of A_b. By uniqueness of the mub of A_b below x, we get $b'=b$. Hence $y\geq B$ and $y=x$.

($\Leftarrow$) Let $x\geq A\subseteq_{fin} D$. By property m there exists $a\in MUB(A)$ s.t. $a\leq x$. Let a' be s.t. $x\leq a'\in MUB(A)$. By applying m again, there exists $b\in MUB(\{a,a'\})$ s.t. $b\leq x$. Since $U^*(A)=U(A)$, we have $b\in U(A)$, i.e. $b\in MUB(A')$ for some $A'\subseteq A$. A fortiori $a,a'\in UB(A')$, hence $a=b=a'$. This proves the uniqueness of A and ends the proof. □

Proposition L-CCC

The category of L-cpos and continuous functions is cartesian closed. The full subcategory **L** of algebraic L-cpos is cartesian closed.

Proof

Let D and E be L-cpos. Suppose that $f,g\leq h$ are in $D\to E$. Then $f(x),g(x)\leq h(x)$. Define k(x) as the minimum upper bound of f(x), g(x) under h(x). This function k is the minimum upper bound of f, g under h (to check the continuity of k, given Δ, one works in $\downarrow h(\sqcup D)$). If D and E are algebraic, then we already know that any h is the lub of the set of step functions below it (cf. chpt. 1, sect. 5)[5]. We have to check that this set is directed. This follows from the consistent completeness of $\downarrow h$. □

As a last result in this section we show that the terminal object, products, and exponents in a full subcategory of **Dcpo**, if any, must be those of **Dcpo**.

Proposition *(terminal object, product, and exponent are pre-determined)*

Let **C** be a full subcategory of **Dcpo**. We denote by $\times$, $\to$ the product and the exponent in **Dcpo**. We write $D\cong E$ when D and E are isomorphic in **Dcpo**, which amounts to D and E being isomorphic in the category of partial orders. Then the following holds.

(1) If **C** has a terminal object T, then T is a one point cpo.

(2) If **C** has a terminal object T and products $D\otimes E$, then $D\otimes E \cong D\times E$

(3) If **C** has terminal object T, finite products, and exponent E^D then: $E^D \cong D\to E$.

[5]This is where we need the restriction to cpos rather than dcpos.

Proof

(1) If T is terminal and has distinct elements x, y, then the constant functions $z \mapsto x$, $z \mapsto y$: $T \to T$ are continuous and distinct: contradiction. We can in the sequel freely confuse $x \in D$ and $x: T \to D$.

(2) Let $D, E \in \mathbf{C}$. Consider the products

- $(D \otimes E, p: D \otimes E \to D, q: D \otimes E \to E)$ in **C**, with pairing denoted by $\prec , \succ$,
- $(D \times E, \pi, \pi')$ in **Cpo**, with pairing $< , >$.

We show that $<p, p'>: D \otimes E \to D \times E$ is an isomorphism in **Cpo**. It is enough to show that

- $<p, p'>$ is *injective*. We have, for any $x, x': T \to D \otimes E$: $<p, p'> \circ x = <p, p'> \circ x' \Leftrightarrow p \circ x = p \circ x$ and $p' \circ x' = p' \circ x' \Leftrightarrow <p, p'> \circ x = <p, p'> \circ x' \Leftrightarrow x = x'$.

- $<p, p'>$ is *surjective*. Let $(y, z) \in D \times E$. We show: $(y, z) = <p, p'>(\prec y, z \succ)$, which is clear since $p(\prec y, z \succ) = y$ and $p'(\prec y, z \succ) = z$.

- If $(y, z) \leq (y', z')$, then $\prec y, z \succ \leq \prec y', z' \succ$. We can assume the existence of an object $C \in \mathbf{C}$ containing at least two elements c, c', s.t. $c < c'$: indeed, if **C** only admits objects of cardinality one then the proposition is trivially true, and if **C** contains only discretely ordered sets, then in particular D, E are discretely ordered, and so are $D \otimes E$ (as an object of **C**) and $D \times E$ (by the definition of product in **Dcpo**). With this fixed C, we can build, for any D, and x, $x' \in D$ s.t. $x \leq x'$, a continuous map $f_{x,x'} : C \to D$ as follows:

$f_{x,x'}(y) = x$ if $y \leq c$
$f_{x,x'}(y) = x'$ otherwise.

With the help of these functions, we have:

$\prec y, z \succ = \prec f_{y,y'}, f_{z,z'} \succ \circ c$, $\prec y', z' \succ = \prec f_{y,y'}, f_{z,z'} \succ \circ c'$.

Thus, by monotonicity of $\prec f_{y,y'}, f_{z,z'} \succ$, we get $\prec y, z \succ \leq \prec y', z' \succ$.

(3) Given (1) and (2), we may work directly with the standard product $\times$. Consider the exponents:

- $(D \Rightarrow E, \varepsilon: (D \Rightarrow E) \times D \to E)$ in **C**, with currying denoted by Λ',
- $(D \to E, ev)$ in **Cpo**, with currying Λ.

We show that $\Lambda(\varepsilon): D \Rightarrow E \to (D \to E)$ is an iso.

- $\Lambda(\varepsilon)$ is *injective*: If $\Lambda(\varepsilon)(h) = \Lambda(\varepsilon)(h')$, then $\forall d \in D.\ \varepsilon(h, d) = \varepsilon(h', d)$, by definition of currying in **Cpo**. This can be rewritten as $\varepsilon \circ (h \times id) = \varepsilon \circ (h' \times id)$. This entails $h = \Lambda'(\varepsilon \circ (h \times id)) = \Lambda'(\varepsilon \circ (h' \times id)) = h'$.

- $\Lambda(\varepsilon)$ is *surjective*. Let $f \in D \to E$. We show: $f = \Lambda(\varepsilon)(\Lambda'(ev)(f))$, which is clear since $\Lambda(\varepsilon) \circ \Lambda'(ev) = \Lambda(\varepsilon \circ (\Lambda'(ev) \times id)) = \Lambda(ev) = id$.

- Finally we show that if $g \leq g'$, then $\Lambda'(ev)(g) \leq \Lambda'(ev)(g')$. Consider $f_{g,g'}: C \to (D \to E)$. We have

$\Lambda'(ev)(g)=\Lambda'(ev\circ(g\times id))=\Lambda'(ev\circ((f_{g,g'}\circ c)\times id))=$
$\Lambda'(ev\circ(f_{g,g'}\times id)\circ(c\times id))=\Lambda'(ev\circ(f_{g,g'}\times id))(c)$

Let $k\triangleq\Lambda'(ev\circ(f_{g,g'}\times id))$. Then we have $\Lambda'(ev)(g)=k(c)$ and $\Lambda'(ev)(g')=k(c')$, whence the result. □

3. The Two Maximal Cartesian Closed Full Subcategories of Acpo

This section is devoted to Jung's classification theorem for algebraic dcpo's, and to Smyth's characterization theorem for ω-algebraic dcpos.

Both L-cpos and bifinite domains satisfy property m. We shall first prove that this property is necessary. Actually we prove that bicompleteness is necessary (which is stronger). Recall (cf. Exercise BICOMPLETE of chapter 1) that a partial order r $(D,\leq)$ is *bicomplete* if both D and $D^{op}=(D,\geq)$ are directed complete.

We need the following result, which is not well-known yet powerful (see Markowsky[76]). Recall that a partial order is well-ordered when every non-empty subset has a minimum (in particular a well-ordered partial order is total, and well-founded, i.e. it admits no infinite strictly decreasing chain).

Fact *(Ordinal completeness)*

Let D be a partial order. D is a dcpo if and only if any well-ordered subset of D has a lub.

We are ready for the first key proposition of the section.

Proposition CONT-BICOMP

A continuous dcpo D with continuous function space is bicomplete.

Proof

The proof is by contradiction. By Fact *Ordinal completeness*, we may assume that there exists an op-well-ordered subset B of D which has no glb. Let A be the set of lower bounds of B. Define the function r on D by

$r(x) = x$ if $x\in A$,
$r(x) = \sqcap\{b\in B \mid b\geq x\}$ if $x\notin A$ (where the glb is meant in B).

This is well defined: $\sqcap\{b\in B \mid b\geq a\}$ is the maximum of the set C of lower bounds in B of $\{b\in B \mid b\geq x\}$, so we only have to check that C is not empty. Since $x\notin A$, we have $\neg(x\leq b')$ for some $b'\in B$. A fortiori, if $b\geq x$, then $\neg(b\leq b')$. But B is a total order, thus $b'<b$, which proves $b'\in C$. Moreover, $r(D)\subseteq A\cup B$, and r is the identity on $A\cup B$. We check that r is continuous. We leave the easy check that r is monotone to the reader. If $\sqcup\Delta\in A$, then $\Delta\subseteq A$, hence $r(\sqcup\Delta)=\sqcup\Delta=\sqcup(\Delta)$. If $\sqcup\Delta\notin A$, then $\neg(\delta\leq b')$ for some $b'\in B,\delta\in\Delta$, i.e. $\delta\notin A$. Hence $\Delta'\cap A=\emptyset$, where $\Delta'=\Delta\cap\delta\uparrow$, and $r(\Delta')\subseteq B$. Clearly $\sqcup\Delta'=\sqcup\Delta$ and $\sqcup r(\Delta')=\sqcup r(\Delta)$. Hence it is enough to prove $r(\delta')=r(\sqcup\Delta')$. A well-ordered set is well-founded, hence by Fact FΔM of chapter 1, $r(\Delta')$ has a maximum $r(\delta')$ for some

$\delta' \in \Delta'$. We have to prove $r(\delta') \geq r(\sqcup\Delta')$, i.e., unfolding the definition of r:

$\forall b' \leq \{b \in B \mid b \geq \sqcup\Delta\}\ \forall b'' \geq \delta'\ \ b' \leq b''$.

We proceed by contradiction. If $\neg(r(\delta') \geq r(\sqcup\Delta'))$, then $\exists b', b''\ \delta' \leq b'' < b' \leq \{b \in B \mid b \geq \sqcup\Delta\}$. Since for any $\delta \in \Delta$ we have $r(\delta) \leq r(\delta')$, we have $\forall \delta\ \neg(b' \leq \{b \in B \mid b \geq \delta\})$, i.e. (since B is a total order) $b' > b_\delta \geq \delta$ for some b_δ. Since B is op-well-ordered, the non-empty set $\{b_\delta \mid \delta \in \Delta\}$ has a maximum $b_{\delta''}$ for some $\delta'' \in \Delta$. We have $b' > b_{\delta''} \geq \sqcup\delta$, which contradicts $b' \leq \{b \in B \mid b \geq \sqcup\Delta\}$. Hence r is continuous. We know from Exercise →RETRACT that $D' \to D'$ is a retract of $D \to D$, where $D' = r(D) = A \cup B$. It is continuous, by Proposition *Retracts of continuous dcpos*. The proof now proceeds via the following claim:

$\exists\ f \in D' \to D'\ f \ll id$ and $f(B) \subseteq B$.

We first prove that the claim contradicts the continuity of $D' \to D'$. Since B is op-well-ordered, it has a predecessor function pred: pred(b) is the maximum b' s.t. $b' < b$ (there is at least one such b', otherwise b would be a minimum of B, contradicting our assumption on B). In particular, $A \cap B \neq \emptyset$, since:
$\forall b \in B\ \neg(pred(b) \geq b)$. Define, for each $b \in B$, a function $g_b : D' \to D'$ by

$g_b(x) = pred(f(x))$ if $x \in B$ and $x \leq b$, $g_b(x) = x$ o.w..

We shall prove (second claim):

- g_b is continuous for all $b \in B$,
- $\{g_b \mid b \in B\}$ is directed and has id as lub,
- there is no g_b s.t. $f \leq g_b$.

The second claim contradicts $f \ll id$. Thus the proof will be finished if we prove the two claims.

We prove first the last part of the second claim. If $f \leq g_b$, then $f(b) \leq g_b(b) = pred(f(b))$, a contradiction to the definition of pred. We prove that $\{g_b \mid b \in B\}$ is actually a chain by proving $b' \leq b \Rightarrow g_b \leq g_{b'}$. The only interesting case is when $x \in B$ and $b' < x \leq b$. Then $g_b(x) = pred(f(x)) \leq f(x) \leq x = g_{b'}(x)$. The equality $id = \sqcup\{g_b \mid b \in B\}$ follows from the remark that $g_{pred(b)}(b) = b$ for all $b \in B$. Finally, we check the continuity of g_b. Let Δ be directed in D'. We have $\sqcup\Delta \in A$ iff $\Delta \subseteq A$, since A is defined as the set of lower bounds of B. The interesting case is $\sqcup\Delta \in B$. Then $\delta \in B$ for some $\delta \in \Delta$. We can choose δ to be the maximum of $B \cap \Delta$, since B is well-ordered. Then $\sqcup\Delta = \delta \in \Delta$, hence the continuity of g_b follows from its monotonicity, which is fairly obvious. This settles the second claim.

As for the first claim, it is obvious if A is empty, since then B=D'. If A is not empty, we know that it has no maximum by the assumption on B. This entails that $A' = \{x \mid \exists y \in A\ x \ll y\}$ is not directed (by continuity of D, the lub of A' would be larger than any element of A, and still belong to A). Hence there exists $x'' \ll x \in A$ and $y'' \ll y \in A$ s.t. x'' and y'' have no upper bound in A'. Let x' and y' be obtained by interpolation: $x'' \ll x' \ll x$ and $y'' \ll y' \ll y$ (in particular $x', y' \in A$). Then x' and y' have no upper bound in A. Since $id = \sqcup\{f \mid f \ll id\}$, and since $x' \ll x = \sqcup\{f(x) \mid f \ll id\}$, we have $x' \leq g(x)$ for some $g \ll id$. Similarly $y' \leq h(y)$. Let f be an upper bound of g, h in $\{f \mid f \ll id\}$.

Then $x' \leq f(x)$ and $y' \leq f(y)$. Let b be an element of B. Then b is an upper bound of x and y, since $x,y \in A$. Hence $f(b) \geq f(x) \geq x'$. Similarly $f(b) \geq y'$, which entails $f(b) \in B$ since the only upper bounds of x' and y' are the elements of B. This completes the proof of claim 1 and the proof of the proposition. □

Exercise →CONT⇒CONT. The hypotheses of the previous proposition are actually redundant. Show that a dcpo with continuous function space is continuous.

Here is the second of the three propositions which lead to the classification theorem.

Proposition →ALG+M⇒BIF

Let D be a dcpo with algebraic function space and s.t. D_o satisfies property M. Then D is bifinite.

Proof

We have to prove, for any finite $A \subseteq D_o$, that $U^*(A)$ is finite. Suppose not. Then for each n, one can find an element in $U^{n+1}(A) \setminus U^n(A) = B^{n+1}$ (we set $A = B^0$). We construct a tree in the following way. The nodes are finite sequences $b_n \ldots b_0$ where $b_i \in B^i$ for all i, and where, for each $i<n$, b_i belongs to a subset of $U^i(A)$ of which b_{i+1} is a mub. The root is the empty sequence, the father of $b_n \ldots b_0$ is $b_{n-1} \ldots b_0$. By construction, this is a finitely branching tree. We show that it is infinite by showing that for any $b \in U^*(A)$ there exists a node $b_n \ldots b_0$ s.t. $b = b_n$. Let n be the minimum s.t. $b \in U^n(A)$, hence $b \in B^n$. By definition of $U^n(A)$ we can find a subset B of $U^{n-1}(A)$ of which b is a mub. If B is also a subset of $U^{n-2}(A)$, then we would not have $b \in B^n$. Hence we can build $b_{n-1}, \ldots, b_0$ as desired. Since the tree is infinite and finitely branching, it has an infinite branch by König's Lemma, which amounts to say that there exists an infinite strictly increasing sequence (b_n) in $U^*(A)$. We keep this in reserve and start now to use the algebraicity of D→D. We have $id = \bigsqcup\{f \mid f \text{ is compact and } f \leq id\}$. In particular $a = \bigsqcup\{f(a) \mid f \text{ is compact and } f \leq id\}$ for a compact implies $a = f(a)$ for some f. Hence by directedness we can find a compact $f \leq id$ for which $\forall a \in A\ a = f(a)$. We prove $\forall a \in U^*(A)\ a = f(a)$. Suppose that we know $\forall a \in U^n(A)$ $a = f(a)$. Let a be a mub of $A' \subseteq U^n(A)$. Then $a \geq f(a) \geq A'$ implies $f(a) = a$. In particular f fixes all the elements of the sequence (b_n) . By continuity if also fixes $\bigsqcup b_n = c$. We shall get a contradiction by proving the following claim:

Claim : If D is a dcpo which has a continuous function space, and if $f \ll id$, then for all d, $f(d) \ll d$.

If the claim is true, then $c = f(c) \ll c$, hence c is compact. But a lub of a strictly increasing sequence is not compact: contradiction.

We prove the claim by appealing again to the "retract" trick already used in the proof of Proposition COMP-BICOMP. Let Δ be s.t. $d \leq \bigsqcup \Delta = e$. Since $\downarrow e = D'$ is a retract of D (cf. section 1), then D'→D' is continuous, as a retract of D→D. We show that $f \ll id$

also holds in D'→D' (notice that since f≤id, f maps D' into D'). For this, it is enough by Lemma =WAY-BELOW to consider a directed Δ'⊆D'→D' s.t. id=⊔Δ' in D'→D'. Each g in Δ' can be extended to D by setting

g(x)=x whenever ¬(x≤e).

Hence Δ' can be viewed as a directed subset in D→D, and has clearly id as lub there too. Hence f≤g for some g of Δ', and the inequality holds a fortiori in D'→D'. We have proved f≪id in D'→D'. Consider the family of constant functions d↦δ for each δ∈Δ. It forms a directed set with lub the constant d↦e. In D'→D' we have d↦e ≥ id. Hence f≤(d↦δ) for some δ∈Δ. In particular f(d)≤δ. This finishes the proof of the claim and the proof of the proposition. □

The following is the key result of Jung[88].

Proposition L or M

Let D and E be algebraic cpos satisfying property m. If D→E is continuous, then E is a L-cpo or D_o satisfies property M.

Proof

Since E is not a L-cpo, then by Proposition L-CPOS there exists c in E, two compact lower bounds a_1 and a_2 of c, and two distinct mubs of $\{a_1,a_2\}$ below c. Since D has property m, D_o has also property m by Lemma *Mubs of compacts* . Since D_o has property m but not M, by Lemma PLOTKIN-2-PLOTKIN there exist x_1 and x_2 in D_o s.t. MUB($\{x_1,x_2\}$) is finite. Assume moreover that D→E is continuous. Then we define g:D→E by

$g(d) = \perp$ if ¬(d≥x_1) and ¬(d≥x_2) ,
$g(d) = a_1$ if d≥x_1 and ¬(d≥x_2) ,
$g(d) = a_2$ if ¬(d≥x_1) and d≥x_2 ,
$g(d) = b_1$ if d≥x_1 and d≥x_2 .

We leave the reader check that g is continuous and is a mub of the step functions $x_1 \to a_1$ and $x_2 \to a_2$. By Lemma *Mubs of compacts* , g is compact. We shall contradict the compactness of g. We define f by replacing b_1 by c in the last equation. Clearly g≤f. We shall exhibit a directed set of functions which has f as lub, but none of which dominates g.

For each finite subset of MUB($\{x_1,x_2\}$), define a function f_A:D→E by

$f_A(d) = \perp$ if ¬(d≥x_1) and ¬(d≥x_2) ,
$f_A(d) = a_1$ if d≥x_1 and ¬(d≥x_2) ,
$f_A(d) = a_2$ if ¬(d≥x_1) and d≥x_2 ,
$f_A(d) = b_2$ if d∈MUB($\{x_1,x_2\}$)\A ,
$f_A(d) = c$ otherwise .

We have to check that the f_A's are continuous, and form a directed set with lub f. We leave this to the reader (to prove the continuity, observe that (x_1,x_2 are compact, ⊔Δ∈MUB($\{x_1,x_2\}$) ⇒ ⊔Δ has a maximum)). Suppose g≤f_A for some A, and

pick $d \in MUB(\{x_1,x_2\}) \setminus A$. We should have $a_2 = g(d) \leq f_A(d) = b_2$, but this does not hold. This finishes the proof by contradiction. □

Theorem 2 Max $\subseteq$ **Acpo**

The categories **Bif** and **L** are the two maximal cartesian closed full subcategories of **Acpo**.

Proof

We have proved that **Bif** and **L** are cartesian closed in section 2. We also proved in section 2 the Proposition *Expected Structure*. Hence, if **C** is a cartesian closed full subcategory of **Acpo**, we know that the exponentials of **C** are the exponentials of **Cpo**. Let D be an object of **C**; We first prove that D is an object of **L** or an object of **Bif**. Since both D and $D \to D$ are algebraic, D is bicomplete by Proposition CONT-BICOMP, hence it has property m (cf. Exercise BICOMPLETE of chapter 1). We can apply Proposition L or M to D and D. Thus D is an algebraic L-cpo, or D_o has property M. If D_o has property M, then by Proposition $\to$ALG+M$\Rightarrow$BIF D is bifinite. Suppose now **C** is a a subcategory of neither **L** nor **Bif**. Then there is an object D of **C** which is not bifinite and an object E of **C** which is not a L-cpo. Yet $D \to E$ is an object of **C**, hence it is continuous, and by Proposition L or M, D_o has property M. Since $D \to D$ is algebraic, D is bifinite by Proposition $\to$ALG+M$\Rightarrow$BIF: contradiction.This concludes the proof. □

The analysis is simplified in the case of ω-algebraic cpos, thanks to the following proposition.

Proposition $\to$ω-ALG$\Rightarrow$M

If D is an (algebraic) cpo and $D \to D$ is ω-algebraic, then D_o has property M.

Proof

We already know from previous proofs that D is bicomplete, hence that D_o has property m. Assume that $MUB(\{x_1,x_2\})$ is infinite for some compact x_1, x_2. We shall build uncountably many mubs of $x_1 \to x_1$ and $x_2 \to x_2$. Since they are compact, this will contradict the ω-algebraicity of D. We pick two distinct mubs b_1, b_2 of x_1, x_2. For any $S \subseteq MUB(\{x_1,x_2\})$, we define $f_S: D \to E$ by

$f_S(d) = \bot$ if $\neg(d \geq x_1)$ and $\neg(d \geq x_2)$,
$f_S(d) = a_1$ if $d \geq x_1$ and $\neg(d \geq x_2)$,
$f_S(d) = a_2$ if $\neg(d \geq x_1)$ and $d \geq x_2$,
$f_S(d) = b_1$ if $\exists s \in S\ d \geq s$,
$f_S(d) = b_2$ if $\exists s \in MUB(\{x_1,x_2\}) \setminus S\ d \geq s$.

f_S is well-defined since D is a L-cpo by Proposition L or M: if $d \in UB(\{x_1,x_2\})$, there is exactly one mub of x_1, x_2 below d. We leave the reader check the rest. □

Exercise: Proposition $\to$ω-ALG$\Rightarrow$M allows to prove Smyth's result. Show that the category **ω-Bif** of ω-bifinite cpos and continuous functions is the largest cartesian

closed full subcategory of **ω-Acpo**.

4. The Four Maximal Cartesian Closed Full Subcategories of Adcpo

In this section, we outline the results of A. Jung in the case of algebraic dcpos. There are four maximal cartesian closed full subcategories of **Adcpo**. The duplication w.r.t. to the previous section comes from the following discriminating proposition, which is "orthogonal" to the discriminating Proposition L or M.

Proposition F or U

Let D and E be continuous dcpo's. If $D \to E$ is continuous, then D has finitely many minimal elements or E is a disjoint union of cpos.

Proof

Suppose that e_1 and e_2 are minimal in E and have an upper bound e, and that D has infinitely many minimal elements. By property m one can find a mub e of e_1, e_2. The constant function $x \mapsto e_1$ is minimal, hence compact in $D \to E$ (minimal implies compact in a continuous dcpo). For any finite set of minimal elements of D, we define a function f_A by

$f_A(d) = e$ if $x \in \uparrow A$,
$f_A(d) = e_2$ otherwise .

We leave the reader check that this defines a directed family of continuous functions which has $x \mapsto e$ as lub (the monotonicity of f_A follows from $e_2 \leq e$). Hence one must have $x \mapsto e_1 \leq f_A$ for some A, which entails $e_1 \leq e_2$. But e_2 is minimal and $e_1 \neq e_2$: contradiction. □

By going to the second order, the first branch of the alternative (finitely many minimal elements) can be strengthened. We call the *root* of a dcpo D the set $U^*(\varnothing)$.

Proposition F-F∞

Let D be a (continuous) dcpo s.t. $(D \to D) \to (D \to D)$ is continuous. Then either D has a finite root or D is a disjoint union of cpos.

Proof

If D is not a disjoint union of cpos, then $D \to D$ is not a disjoint union of cpos (take two minimal elements e_1 and e_2 in D which have an upper bound e, and take the corresponding constant functions in $D \to D$). We know from Proposition CONT-BICOMP that $D \to D$ is bicomplete, hence satisfies property m, and from Proposition F-F∞ that $D \to D$ has finitely many minimal elements. We shall show that this entails the finiteness of the root of D. With each element d of the root we associate the canonical retraction r_d onto $\downarrow d$ defined in Lemma *(Simple retractions)*. Any mapping $f \leq r_d$ must fix $\downarrow d$ (cf. the proof of Proposition

→ALG+M⇒BIF). We show that if $d \neq d'$, r_d and $r_{d'}$ have no common lower bound. We can assume say $\neg(d \leq d')$. Then if $f \leq r_d, r_{d'}$, we have

$$f \leq r_d \Rightarrow f(d)=d \ , \ f \leq r_d \Rightarrow f(d) \leq r_{d'}(d)=d',$$

hence $d=f(d) \leq d'$, contradicting the assumption. Since D→D satisfies property m, there exists a minimal function m_d below each r_d. The m_d's are all distinct, since $m_d=m_{d'}$ would entail that r_d and $r_{d'}$ have a lower bound. Hence if the root of D is infinite, then D→D has infinitely many minimal elements: contradiction. □

The last key point is the following lemma which shows that the results of the previous section can be exploited.

Lemma ∀L or ∀M

Let D and E be algebraic dcpos satisfying property m. If ↑e is not a L-cpo and if $(\uparrow d)_o$ has not property M, for some compacts e, d of E, D respectively, then D→E is not continuous.

Proof

Obvious consequence of Proposition L or M and Exercise →RETRACT. □

Here is Jung's classification theorem:

Theorem 4 Max ⊆ **Adcpo**

There are four maximal cartesian closed full subcategories of **Adcpo**, the objects of which are, respectively:

- the disjoint unions of algebraic L-cpos,
- the disjoint unions of bifinite cpos,
- the dcpos with a finite root and s.t. all their Scott-open basic opens ↑d are algebraic L-cpos,
- the profinite dcpos.

Proof

Hint: The proof proceeds like the proof of Theorem 2 Max ⊆ **Acpo**, exploiting not only the discrimination L or M (in its variant ∀L or ∀M), but also the discrimination F or U. One shows that the profinite dcpos are the dcpos with a finite root s.t. all ↑d's are bifinite. □

Exercise: Show that the category **ω-Prof** of ω-profinite dcpos and continuous functions is the largest cartesian closed full subcategory of **ω-Adcpo**.

Hint for exercise →CONT⇒CONT: Use the claim proved in Proposition →ALG+M⇒BIF.

4. Monads and Computations

Contents: 1. Representing Computations as Monads, 2. Partial Maps and the Monad of Partial Computations, 3. An Adequacy Proof for a Call-by-value PCF, 4. Control Operators and CPS Translations, 5. Monads of Powerdomains.

In the analysis of programming languages and logics it is often helpful to distinguish between the notion of *value* and the notion of *computation*.

A first example coming from recursion theory relies on the notions of total and partial map. In our jargon a total map when given a *value* always returns a *value* whereas a partial map when given a *value* returns a possibly infinite *computation*. This example suggests that (i) the denotation of a partial recursive algorithm is a map from *values to computations*, and (ii) that values are particular kinds of computations.

In domain theory the divergent computation is represented by a bottom element, "$\perp$", that we add to the collection of values. This can be seen as the motivation for the shift from sets to flat domains (cf. discussion in chpt. 1).

In the models of typed (and type-free) λ-calculus that have been presented so far this distinction was forgotten by regarding "$\perp$" as an element with the same status of a value, hence the denotation of a λ-term is a map from values to values. There are however certain paradigms for the evaluation of λ-expressions that do keep this distinction, notably the eager and the lazy evaluation. Roughly in the eager evaluation we first evaluate the argument of an application hence the denotation of a program can be seen, as in recursion theory, as a map from values to computations. In the lazy evaluation the argument of an application is frozen into a program and its environment (i.e. a computation), hence the denotation of a program can be seen as a map from computations to computations.

Another framework where the distinction between values and computations is useful is that of fixpoints extensions of typed λ-calculi. Consider for example a simply typed λ-calculus and its "Curry-Howard correspondence" with the minimal propositional logic of implication.[6] Suppose that we want to enrich the calculus with a fixed point combinator Y, on terms, entailing fully recursive definitions. Which *type* should we assign to Y ? One possibility is to introduce a family of combinators Y_α of type $(\alpha\to\alpha)\to\alpha$ (cf. chpt. 1). Then the correspondence with the logic is blurred as $Y_\alpha(\lambda x{:}\alpha.x)$ has type α for any type α, i.e. every type is inhabited. Another possibility is to regard $Y_\alpha(\lambda x{:}\alpha.x)$ as a *computation of a proof*, that is assign to Y_α the type $(c(\alpha)\to c(\alpha))\to c(\alpha)$, where $c(\alpha)$ is the type representing the computations over α. Then, at the cost of a complication of the formal system, we may keep a correspondence between propositions and a *subset* of types.

[6] According to this correspondence types can be seen as propositions and lambda terms can be seen as proofs.

In these examples we have roughly considered computations as values enriched with an element denoting the divergent computations. There are however other possible notions of computations that arise in the study of programming languages. For instance if we wish to model *non-determinism* then a computation may consist of a collection of values representing the possible outcomes of a program.

Which are then the common properties of these notions of computation ? The notion of *monad* that we describe in the next section seems to provide a good general framework.

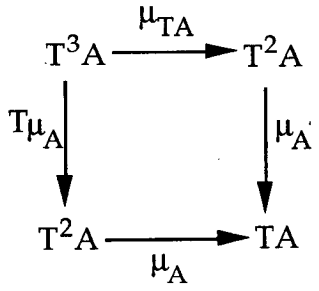
1. Representing Computations as Monads

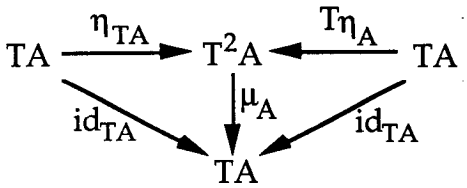
In this section, following Moggi[89], we present the notion of computation-as-monad. The monads of partial computations and the monads of powerdomains will be the leading and motivating examples.

The notion of monad (or triple) is an important category theoretic notion, we refer to Barr&Wells[85] and MacLane[71] for the basic information. What is important here is to state which are the basic computational properties we wish to formalize. Suppose that **C** is our category of data types. An endofunctor T: **C**→**C** defines how to go from a certain collection of values to the computations over such values. A natural transformation η: $Id_C \to T$ determines how a value can be seen as a computation. Another natural transformation μ: $T^2 \to T$ explains how to flat down a computation of a computation to a computation. These requirements plus certain natural commutation properties are expressed in the following

Definition *(monad)*

A monad over a category **C** is a triple (T, η, μ) where T: **C**→**C** is a functor, η: $Id_C \to T$, μ: $T^2 \to T$ are natural transformations and the following diagrams commute:





A monad *satisfies the mono requirement* if η_A is a mono, for any object A.

Examples

We give three basic examples of monads with a computational flavour in the category of Sets. We leave to the reader the needed verifications.

- **Partial Computations.** Define $(\ )_\perp$: Set→Set as:

  $(\ )_\perp(X) \triangleq X \cup \{\perp_X\}$, where, by convention $\perp_X \notin X$.

  $(\ )_\perp(f)(z) \triangleq$ if $z \in X$ then $f(x)$ else $\perp_Y$, where $f: X \to Y$.

  $\eta_X(x) \triangleq x$. $\mu_X(x) \triangleq$ if $z \in X$ then x else $\perp_X$.

- **Non-deterministic Computations.** Define P: Set→Set as:

  $P(X) \triangleq P_{fin}(X)$. $P(f)(a) = f(a)$, where $f: X \to Y$.

  $\eta_X(x) \triangleq \{x\}$. $\mu_X(z) \triangleq \cup z$.

- **Continuations.** Let be given a set of "results", R, containing at least two elements. In order to understand the basic trick behind the notion of computation one should think of the "double negation" interpretation of classical logic into intuitionistic logic. Define C: Set→Set as:

  $C(X) \triangleq (X \to R) \to R$.

  $C(f) \triangleq \lambda g \in (X \to R) \to R.\ \lambda h \in (Y \to R).\ g(h \circ f)$, where $f: X \to Y$.

  $\eta_X(x) \triangleq \lambda h \in (X \to R).\ h(x)$.

  $\mu_X(H) \triangleq \lambda h \in (X \to R).\ H(\ \lambda g \in (X \to R) \to R.\ g(h)\ )$.

In the next two sections we will consider in detail the monad of partial computations, for the time being let us concentrate on the monads of non-deterministic computations and continuations. We now introduce two variants of the imperative language studied in 1.1 and analyse their interpretations in a suitable monad (for the sake of simplicity we leave out recursion and expressions).

L_N: s::= a | dummy | s;s | s+s

L_C: s::= a | dummy | s;s | stop

In L_N we have introduced an operator '+' for the non-deterministic composition of two statements. The intuition is that the statement s_1+s_2 can choose to behave as either s_1 or s_2. It is then natural to consider the interpretation of a statement as a morphism from S to $P_{fin}(S)$, where S is the collection of states. Hence, using the monad of non-deterministic computations one defines:

$$[\![dummy]\!] = \eta_S \qquad [\![a]\!] = \eta_S \circ \underline{a}\ , \quad \text{for } \underline{a}: S \to S$$

$$[\![s_1;s_2]\!] = \mu_S \circ P_{fin}([\![s_2]\!]) \circ [\![s_1]\!] \qquad [\![s_1+s_2]\!] = \lambda\sigma.\ [\![s_2]\!]\sigma \cup [\![s_1]\!]\sigma$$

In L_C we have introduced a statement stop whose intuitive effect is that of terminating immediately the execution of a program and return the current state. It appears that the interpretation given in 1.1 is not adequate to interpret this kind of commands which alter in some global way the flow of the control. For instance we should have:

$[\![stop;s]\!] = [\![stop]\!]$, for any s

which is hopeless if we insist in stating $[\![stop;s]\!] = [\![s]\!] \circ [\![stop]\!]$. However in the same section we have learned that the notion of continuation is useful in modelling control. Before proceeding it is worth pointing out that in the following interpretation the 'type of the domain' where statements are interpreted differ

from the one in 1.1 since: (i) we remove the environment (we do not need it here), and (ii) we consider an isomorphic representation to fit the 'monadic' point of view, namely we use: $(S\to S)\to(S\to S) \cong S\to C(S) = S\to((S\to S)\to S)$. Hence:

$$[\![dummy]\!] = \eta_S \qquad [\![a]\!] = \eta_S \circ \underline{a}\,, \quad \text{for } \underline{a}: S \to S$$

$$[\![s_1;s_2]\!] = \mu_S \circ P_{fin}([\![s_2]\!]) \circ [\![s_1]\!] \qquad [\![stop]\!] = \lambda\sigma.\, \lambda f.\, \sigma$$

Exercise: Verify $[\![a;b]\!] = \lambda\sigma.\, \lambda f.\, f(\underline{b}(\underline{a}\sigma))$.

An easy remark is that the interpretations for L_C and L_N are formally identical but for the fourth clause. As a matter of fact we have been using a general pattern in these interpretations which goes under the name of 'Kleisli category'. Given a monad (T, η, μ) over a category **C** the Kleisli category **K(C)** is formed as follows:

$$Ob_C = Ob_{K(C)}\,, \qquad K(C)[a, b] = C[a, Tb]$$

$$id_a = \eta_a\,, \qquad f \circ g = \mu_c \circ Tf \circ g \qquad \text{if } g: a\to Tb,\ f: b\to Tc \text{ in } \mathbf{C}.$$

The reader will find in Moggi[89] more information on this construction and on its use in the definition of the interpretation of a meta-language where the notion of computation is treated abstractly as a monad with certain desirable properties.

2. Partial Maps and the Monad of Partial Computations

Most of the motivating examples discussed in the introduction rely on the notion of partial computation. In the previous section we have defined the monad of partial computations over **Set**. The main point of this section is to show how the monad of partial computations can be derived by a general notion of *partial map*. We will then apply this connection between partial maps and monads of partial computations to the categories of domains introduced in the previous chapters.

In particular we will introduce the basic notion of *partial cartesian closed category* (pccc). Every pccc has an object Σ, called *dominance*, that 'classifies' the *admissible subobjects* (in the same sense as the object of truth-values Ω classifies *arbitrary* subobjects in a topos). To give a taste of things to come let us consider the familiar category of complete partial orders and continuous maps, **Dcpo**. In **Dcpo** we can choose as admissible monos (i.e. subobjects) the ones whose image is a Scott open. Then the dominance is represented by Sierpinski space $O = \{\bot, \top\}$, the two points cpos. The dominance O *classifies* the admissible monos because any Scott open U over the cpo D determines a unique continuous maps, $f: D\to O$ such that $f^{-1}(\top) = U$.

Admissible Family of Monos[7]

It is standard to consider an equivalence class of monos on an object as a generalized notion of subset. A *partial map* from a to b can then be represented as a total map from a subset of a to b. In many interesting examples the domain of convergence of a partial map is not arbitrary. E.g. it is open (as in **Dcpo**), recursively enumerable, etcetera. It is then reasonable to look for a corresponding categorical notion of admissible mono as specified by the following

Definition *(Admissible family of monos)*

An admissible family of monos M for a category **C** is a collection $\{M(a) \mid a \in C\}$ such that:

(1) If $m \in M(a)$ then m is a *mono* $m: d \rightarrowtail a$.

(2) The *identity* on a is in M(a): $id_a \in M(a)$.

(3) M is closed under *composition* i.e.

if $m_1: a \rightarrowtail b \in M(b)$ and $m_2: b \rightarrowtail c \in M(c)$ then $m_2 \circ m_1: a \rightarrowtail c \in M(c)$.

(4) M is closed under *pullbacks* i.e.

if $m: d \rightarrowtail b \in M(b)$ and $f: a \rightarrow b$ then $f^{-1}(m) \in M(a)$.

The related category of partial maps

An admissible family of monos M on C enjoys properties which are sufficient for the construction of a related category of partial maps pC. A representative for a partial map from a to b is a pair of maps in C, (m, f), such that $m: d \rightarrowtail a \in M(a)$ determines the domain and $f: d \rightarrow b$ the functional behavior. The category pC has the same objects as C and as morphisms equivalence classes of representatives of partial maps, namely:

$$pC[a, b] \triangleq \{[m,f] \mid m: d \rightarrowtail a \in M(a), f: d \rightarrow b\}$$

where (m,f) is equivalent to (m',f') iff they have the same domain and codomain, and there is a map that makes m isomorphic to m' in the slice category C/a, and f isomorphic to f' in the slice category C/b. To specify domain and codomain of a (representative) for a partial map we write $[m,f]: a \rightharpoonup b$ ($(m,f): a \rightharpoonup b$).

Definition *(lifting)*

Given a category enriched with a collection of admissible monos, say (**C**, M) and an object a in **C** the lifting of a is defined as a map, open: $(a)_\perp \rightarrow a$, such that

$$\forall b \in \mathbf{C}.\ \forall f: b \rightarrow a.\ \exists! f': b \rightarrow (a)_\perp .\ f = open \circ f'$$

Given (**C**, M) there is a canonical embedding functor, Emb: $\mathbf{C} \rightarrow \mathbf{pC}$, defined as

$Emb(a) \triangleq a$, $Emb(f) \triangleq [id, f]$.

The following theorem characterizes the lifting as the right adjoint of the embedding functor and shows that it induces a monad.

[7] Refer to Curien&Obtulowicz[88], Moggi[88] and Robinson&Rosolini[88] for extended information on the origins and the development of the theory. Longo&Moggi[84] is the first place were we found the definition of pccc.

Theorem

(1) (**C**, M) has liftings iff the embedding functor has a right adjoint.

(2) The lifting functor induces a monad over **C**.

Proof *(sketch)*

(1) ($\Rightarrow$) Define a lifting functor, Lift: $\mathbf{pC}\rightarrow\mathbf{C}$, as $\text{Lift}(a) \triangleq (a)_\perp$, $\text{Lift}(f) \triangleq (f\circ\text{open}_a)'$, where f: a$\rightarrow$b. Next define a natural iso, τ: $\mathbf{pC}[\_,\_]\rightarrow\mathbf{C}[\_,\text{Lift}\_]$, as $\tau_{a,b}(f) \triangleq f'$.

($\Leftarrow$) Given the natural iso τ define $(a)_\perp \triangleq \text{Lift}(a)$, $\text{open}_a \triangleq \tau^{-1}(\text{id}_{(a)_\perp})$.

(2) Define $\eta_a \triangleq (\text{id}_a)'$, and $\mu_a \triangleq \tau_{(a)_{\perp\perp},a}(\text{open}_a\circ\text{open}_{(a)_\perp})$. □

Exercise pSet: Find a notion of admissible mono in **Set** that generates the monad of partial computations defined in the previous section.

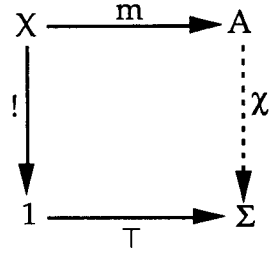
Definition *(Pccc)*

Let M be an admissible collection of monos on the category **C**. (**C**, M) is a *pccc* (partial cartesian closed category) if **C** is *cartesian* and for any pair of objects in **C**, say a, b, there is a pair (pexp(a, b), $\text{peval}_{A,B}$: pexp(a, b)$\times$a$\rightarrow$b) with the universal property that for any f: (c$\times$a)$\rightarrow$b there exists a unique h: C$\rightarrow$pexp(A, B) (denoted $p\Lambda_{a,b,c}(f)$) such that $\text{peval}_{a,b} \circ (h\times\text{id}_a) = f$.

In other words there is a functor *partial exponent* on b pexp_b: $\mathbf{pC}\rightarrow\mathbf{C}$ that is right adjoint to the product functor _$\times$b: $\mathbf{C}\rightarrow\mathbf{pC}$ that is: $pC[\_\times b, \_] \cong C[\_ ,\text{pexp}_b\_]$. By instantiating the natural isomorphism one obtains the following version of "currying": a$\times$b$\rightarrow$c $\cong$ a$\rightarrow$(b$\rightarrow$c), where we write more suggestively b$\rightarrow$c for $\text{pexp}_b(c)$. Indeed by virtue of the previous isomorphism we can safely confuse b$\rightarrow$c with pC[b, c]. Finally the *lifting* can be defined as $(a)_\perp \triangleq 1\rightarrow a$, with the map pev $\circ$ <id, !>.

Fact *(Dominance)*

In every pccc there is an object $\Sigma \triangleq (1)_\perp \triangleq 1\rightarrow 1$, called dominance, that classifies the admissible monos in the following sense: $\exists\top$: $1\rightarrow\Sigma$. $\forall A$. $\forall m\in M(A)$. $\exists!\ \chi$ that makes the following diagram into a pullback



Exercises: (1) Given a partial category define an "admissible subobject" functor M(_) : $\mathbf{C}^{op}\rightarrow$ Set. Show that the classifier condition can be reformulated by saying that there is a natural isomorphism between the functor M(_), and the hom-functor C[_ , Σ].

(2) Show that in a pccc the following isomorphism holds: $a \to \Sigma \triangleq a \to (1)_\perp \cong a \to 1$.

We can conclude this section by giving precise contents to our initial considerations on **Dcpo**.

Definition

Let **Dcpo** be the category of dcpos and (Scott-)continuous maps. We consider the following class of monos in **Dcpo**: $m: D \to\!\!\!\!\!\circ\; E \in M_S \Leftrightarrow im(m) \in \tau_S(E)$.

Proposition

(1) The class M_S is an admissible family of monos for the category **Dcpo**.

(2) The related category of partial maps is a pccc.

Exercises: (1) Show that the partial category generated by (M_S, **Dcpo**) is equivalent to the categories of (i) dcpos and partial continuous maps, and (ii) cpos and strict continuous maps; where a partial continuous map f: D→E is a partial map such that $Dom(f) \in \tau_S(D)$ and $f_{|Dom(f)}$: Dom(f)→E is Scott continuous, and a strict continuous map is a continuous map, f: D→E, such that $f(\perp_D) = \perp_E$.

(2) Calculate the dominance of (M_S, **Dcpo**).

3. An Adequacy Proof for a Call-by-value PCF

In this section we apply the idea of distinguishing between total and divergent computations which is implicit in the monad of 'partial computations' to the design of a variant of PCF calculus. This gives us the opportunity to revisit the general problem of relating interpretations of a programming language with the way such programming language is executed (cf. chpt. 1, section 6).

Given a programming language, say as an (ordered) initial algebra over some signature of types and operators, the *specification of the operational semantics* is given in two steps:

(1) *Evaluation:* a collection of "programs" is defined, usually the collection of closed terms, on which a partial relation of evaluation is defined. The evaluation is intended to describe the dynamic evolution of a program while running on an abstract machine.

(2) *Observation:* a collection of "admissible observations" is given. These observations represent the only mean to record the behavior of the evaluation when applied to a program (note that the evaluation is not defined on arbitrary terms).

In this fashion an "observational equivalence" can be defined on arbitrary terms M and N as follows: M is *observationally equivalent* to N iff whenever M and N can be plugged into a piece of code P[ ], so to form correct programs P[M] and P[N],

then M and N are not separable (or distinguishable) by any legal observation.

On the other hand any interpretation of a programming language provides us with a theory of programs equivalence. How does this theory compare to observational equivalence? We will say that an interpretation (or a model) is *adequate* whenever it provides us with a theory of equivalence which is contained in the observational equivalence.

We can now ask the following question (reversing the historical evolution of the topic): for which kind of simply typed λ-calculus does a *pccc* provide an adequate interpretation? In order to give an answer to this question we have to fix the rules of observation: we will assume that we can only observe the termination of programs evaluation. Next let us consider the definition of the evaluator. There are two critical points: (i) the evaluator has to stop at lambda abstractions, (ii) in an application the evaluator has to diverge if the argument diverges. The main result will now say that the interpretation of a program (i.e. a closed term) is a total morphism iff its evaluation converges. From this, it immediately follows that the interpretation is adequate.

Having explained the general framework, we can now enumerate the main technical points of this section.

- A *language* based on a fixed point extension of the simply typed lambda calculus is introduced.
- A call-by-value *evaluation* mechanism for the closed terms of this language is defined, and it is stipulated that termination at all types can be observed.
- A *semantic structure* for the the language is discussed, which is based on the partially cartesian closed category of directed complete partial orders and partial continuous maps.
- A *standard interpretation* of the language in the semantic structure is described and some of its basic properties are given.
- In *relating evaluation and interpretation*, it is first proved that the evaluation of a closed term converges to a canonical term iff its denotation is a total morphism.
- As a corollary, a result of *adequacy* of the interpretation w.r.t. a natural observational preorder is proved.
- The notion of *adequacy relation* is analysed, as it represents the very basis of the adequacy theorem.

The new techniques introduced in this section, following Martin-Löf[83] and Plotkin[85], concern essentially the last point. The hard part consists in showing that the evaluation of a closed term converges to a canonical form whenever its denotation is a total morphism. This requires the introduction of a family of "adequacy relations"[8] over the types that relates denotations and closed term in a form that combines the "reducibility candidates" technique (introduced by Tait in its proofs of normalization) with the "admissible predicates" technique (developed

[8] We indicate these relations by their use rather than by their properties as the latter form a rather long list subject to variations according to the language.

by Milne and others in the context of proving properties of fixpoints of continuous functionals).

Language

Types and raw terms are defined by the following BNFs:

Type Variables: tv::= t | s...
Types: $\alpha ::= 1 \mid tv \mid (\alpha \rightarrow \alpha)$
Term Variables: v::= x | y | ...
Terms: $M ::= * \mid v \mid (\lambda v{:}\alpha.M) \mid (MM) \mid (Y_\alpha M)$

Contexts Γ are defined as in chpt. 2. Provable typing judgments are inductively defined by the following rules (in the following we often omit the type label from the Y combinator):

(*) $\Rightarrow \Gamma \supset *: 1$
(asmp) $x: \alpha \in \Gamma \Rightarrow \Gamma \supset x: \alpha$
($\rightarrow$I) $\Gamma, x{:}\alpha \supset M{:}\beta \Rightarrow \Gamma \supset (\lambda x{:}\alpha.M): (\alpha \rightarrow \beta)$
($\rightarrow$E) $\Gamma \supset M: (\alpha \rightarrow \beta)$, $\Gamma \supset N{:}\alpha \Rightarrow \Gamma \supset (MN): \beta$
(Y) $\Gamma \supset M: (1 \rightarrow \alpha) \rightarrow \alpha \Rightarrow \Gamma \supset (Y_\alpha M): \alpha$

Note

The types of the Y clause may seem a bit puzzling at a first glance. One can give a semantic justification by recalling that in a pccc one defines the lifting as: $(\alpha)_\perp \triangleq (1 \rightarrow \alpha)$, on the other hand the partial functional space, say $\rightharpoonup$, relates to the total functional space, say $\rightarrow$, as: $\alpha \rightharpoonup \beta = \alpha \rightarrow (\beta)_\perp$. So $M: (1 \rightarrow \alpha) \rightarrow \alpha$ is the "same" as $M: (\alpha)_\perp \rightarrow (\alpha)_\perp$ and the implicit type we are giving to Y is $((\alpha)_\perp \rightarrow (\alpha)_\perp) \rightarrow (\alpha)_\perp$, that is the usual type of a fixed-point combinator over $(\alpha)_\perp$. One good reason to restrict recursion to lifted objects is that these objects have a least element; obviously a continuous function over a directed complete partial order without a least element does not need to have a fix-point.

Evaluation

The canonical forms are the closed, well-typed term that are generated by the following grammar:

$C ::= * \mid (\lambda v{:}\,\alpha.\, M)$

The notion of evaluation "$\mapsto$" is a family of relations, indexed over types, between closed terms and canonical forms. Its definition proceeds by induction on the structure of a well-typed closed term as follows:

(*)	$\Rightarrow$ $*\mapsto*$
(asmp)	"we never evaluate a free variable"
($\rightarrow$I)	$\Rightarrow$ $\lambda x{:}\alpha.M \mapsto \lambda x{:}\alpha.M$
($\rightarrow$E)	$M\mapsto\lambda x{:}\alpha.M'$, $N\mapsto C'$, $[C'/x]M'\mapsto C \Rightarrow MN\mapsto C$
(Y)	$M(\lambda x{:}1.YM)\mapsto C \Rightarrow YM\mapsto C$ (for x fresh variable)

We write $M\downarrow$ if $\vdash M\mapsto C$, for some canonical form C. Note that the definition of "$\mapsto$" gives directly a deterministic procedure to reduce, if possible, a closed term to a canonical form. Canonical forms always reduce to themselves.

Semantic Structure

What suffices to interpret the language is a partially cartesian closed category with "partial fixed points", i.e. with fixed points over lifted objects. Namely

Definition *(partial fixed points)*

A pccc has partial fixed points if for any object A there exists a morphism $pFix_A$: $pexp(pexp(1, A), A)\rightarrow A$, satisfying for any f: $pexp(1, A)\rightarrow A$:

$$pev\circ\langle f,\ pFix_A\circ f\rangle = pFix_A\circ f$$

Proviso

In the following we concentrate on the category of directed complete partial orders and partial continuous maps. Then, as usual, there is a least fixed point operator over lifted objects that is calculated as the join of an inductively defined chain. As we will see this aspect of the semantic structure has an important impact on the definition of adequacy relations. A more abstract framework for an adequacy proof could be given relying on the notion of "O-category" that will be introduced in chpt. 5 (roughly an O-category is a category in which the hom-sets are dcpos and composition is continuous).

Interpretation

Types. We denote with **T** the collection of type interpretations, in our case the collection of dcpos. We are going to give an interpretation that is parametric w.r.t. an assignment η: tv$\rightarrow$**T**. If we consider a pccc whose objects are element in **T** an interpretation of the types is defined by induction as follows:

$[\![1]\!] = \mathbf{1}$	(the terminal object)
$[\![t]\!] = \eta(t)$	
$[\![\alpha\rightarrow\beta]\!] = pexp([\![\alpha]\!], [\![\beta]\!])$	(the partial exponent)

Terms. The interpretation of a judgment $\vdash(x_1{:}\ \alpha_1),\ldots,(x_n{:}\ \alpha_n) \supset M{:}\ \alpha$ is a partial morphism of type $[\![1]\!]\times[\![\alpha_1]\!]\times\ldots\times[\![\alpha_n]\!]\rightarrow[\![\alpha]\!]$ ($\times$ associates to the left). If $\vdash\varnothing \supset M{:}\ \alpha$, that is the term is closed, then we have either a divergent map or a point in $[\![\alpha]\!]$. In the latter case we write $M\Downarrow$.

(*) $[\![\Gamma \supset *: 1]\!] = !_{[\![\Gamma]\!]}$ (1)

(asmp) $[\![(x_1: \alpha_1),\ldots, (x_n: \alpha_n) \supset x_i: \alpha_i]\!] = \pi_{n,i}$

(→I) $[\![\Gamma \supset \lambda x{:}\alpha.M{:}\alpha\to\beta]\!] = p\Lambda([\![\Gamma, x{:}\alpha \supset M{:}\beta]\!])$

(→E) $[\![\Gamma \supset MN{:}\beta]\!] = pev_\circ < [\![\Gamma \supset M{:}\ \alpha\to\beta]\!] , [\![\Gamma \supset N{:}\alpha]\!] >$ (2)

(Y) $[\![\Gamma \supset YM{:}\alpha]\!] = \sqcup_{n<\omega} f(n)$ (3)

where: $g = [\![\Gamma \supset M{:}(1\to\alpha)\to\alpha]\!]$, $f(0) = \uparrow_{[\![\Gamma\to\alpha]\!]}$, $f(n+1) = pev_\circ < g, \underline{id}_\circ f(n) >$

Notes: (1) This is the unique total map into **1**. (2) The operation < , > here is a partial pairing, it is defined only if its arguments are both defined. (3) The morphism $\underline{id}$: A→pexp(**1**,A) is uniquely determined by the identity over A, and a morphism $open_A$: pexp(1,A)→A. Also note that we use here the proviso of working with an O-category.

Lemma *(Substitution)*

If $\vdash \Gamma, x{:}\alpha \supset M{:}\beta$, and $\vdash \Gamma \supset N{:}\alpha$ then

(1) $\vdash \Gamma \supset [N/x]M{:}\beta$

(2) $[\![\Gamma \supset [N/x]M{:}\beta]\!] = [\![\Gamma, x{:}\alpha \supset M{:}\beta]\!]_\circ <id, [\![\Gamma \supset N{:}\alpha]\!]>$

Proof

As in chpt. 2 one proceeds by induction on the length of the typing judgment. □

Adequacy

We want to prove that given a well typed closed term M, $M\Downarrow$ iff $M\downarrow$. It is easy to show that if $M\downarrow$ then $M\Downarrow$ as the interpretation is invariant under evaluation and the interpretation of a canonical form is a total morphism. In the other direction the naive attempt of proving ($M\Downarrow \Rightarrow M\downarrow$) by induction on the typing of M does not work. We are going to associate to every type α an "adequacy relation" R(α) relating denotations and closed terms of type α (cf. chpt. 1). Adequacy relations enjoy the property: $f\ R(\alpha)\ M \wedge f\Downarrow \Rightarrow M\downarrow$, moreover they enjoy additional properties so that a proof by induction on the the typing can go through.

Definition *(adequacy relation)*

A relation $S \subseteq [\![1\to\alpha]\!] \times \Lambda^o{}_\alpha$ is an adequacy relation of type α if it satisfies the following conditions:

(C.1) $(f\,S\,M \wedge f\Downarrow) \Rightarrow M\downarrow$.

(C.2) $(f\,S\,M \wedge \vdash M\mapsto C \wedge \vdash M'\mapsto C) \Rightarrow f\,S\,M'$.

(C.3) $\uparrow_{[\![1\to\alpha]\!]} S\,M$, for any $M\in\Lambda^o{}_\alpha$.

(C.4) $\{f_n\}_{n<\omega}$ directed in $[\![1\to\alpha]\!] \wedge \forall n.\ f_n S\,M \Rightarrow \sqcup_{n<\omega} f_n S\,M$.

We denote with AR(α) the collection of adequacy relations of type α. For any type α, the relation $\{(f, M)\in[\![1\to\alpha]\!] \times \Lambda^o{}_\alpha \mid f\Uparrow\}$, is an adequacy relation of type α.

Note *(adequacy relations are admissible predicates)*

Here we want to emphasize certain *geometric* properties of adequacy relations. In the first place we need to make explicit a cpo structure on the collection of closed terms. Define an equivalence relation, say $\approx$, on terms by stating that $M \approx N$ iff either both terms diverge, or both terms converge to the same canonical form. Given a type α consider the quotient ${\Lambda^o}_\alpha/\approx$, with a *flat order* obtained by assuming that the equivalence class of diverging terms is the least element, and all other equivalence classes are incomparable.

We can now consider $E \triangleq [\![1 \to \alpha]\!] \times {\Lambda^o}_\alpha/\approx$ as the product cpo. By definition a subset $P \subseteq E$ is an *admissible predicate* if it is closed under directed sets. Note that any admissible predicate P determines a relation S_P over $[\![1 \to \alpha]\!] \times {\Lambda^o}_\alpha$ as follows:

$$(f, M) \in S_P \Leftrightarrow (f, [M]_\approx) \in P .$$

Verify that S_P satisfies conditions (C.2) and (C.4), but not vice versa, as one can have a relation S such that: $\neg M\downarrow \wedge \neg M'\downarrow \wedge f\ S_P\ M \wedge \neg f\ S_P\ M'$.

We now put an upper and a lower bound on the collection of admissible predicates we are interested in. The upper bound is $U \triangleq \{(f, [M]_\approx) \mid f\Downarrow \Rightarrow M\downarrow\}$. The lower bound is $L \triangleq \{(f, [M]_\approx) \mid f\Uparrow\}$. Verify that U and L are admissible predicates. Next it is easy to show that the admissible predicates included between L and U are in *bijective* correspondence with the adequacy relations (hint: the upper bound forces (C.1), and the lower bound (C.3)). Hence, to summarize, adequacy relations can be seen as a *particular case* of admissible predicates.

Given an assignment θ: $tv \to \cup_{t \in tv} AR(t)$, such that $\theta(t) \in AR(t)$ for any t, we wish to assign to each type α an adequacy relation of type α.

Definition *(associating adequacy relations to types)*

We associate to a type α a relation $R(\alpha) \subseteq [\![1 \to \alpha]\!] \times {\Lambda^o}_\alpha$ as follows:

$$R(1) \triangleq \{(f, M) \in [\![1 \to 1]\!] \times {\Lambda^o}_1 \mid f\Uparrow \vee (f\Downarrow \wedge M\downarrow)\}$$
$$R(t) \triangleq \theta(t)$$
$$R(\alpha \to \beta) \triangleq \{(f, M) \in [\![1 \to (\alpha \to \beta)]\!] \times {\Lambda^o}_{\alpha \to \beta} \mid (f\Downarrow \Rightarrow M\downarrow) \wedge (d\ R(\alpha)\ N \Rightarrow (pev \circ \langle f, d\rangle\ R(\beta)\ MN))\}$$

Proposition

The relation $R(\alpha)$ is an adequacy relation of type α, for any type α.

Proof

We proceed by induction on the structure of α.

Case (1). By definition of R. *Case (t).* By definition of θ .

Case ($\alpha \to \beta$).

(C.1): By definition of $R(\alpha \to \beta)$.

(C.2): Suppose $(f\ R(\alpha \to \beta)\ M \wedge \vdash M \mapsto C \wedge \vdash M' \mapsto C)$. First observe:

$(\vdash M \mapsto C \wedge \vdash M' \mapsto C \wedge \vdash MN \mapsto C') \Rightarrow \vdash M'N \mapsto C'$ (a).

The interesting case arises if $pev \circ \langle f, d\rangle \Downarrow$. Then we have to show:

$pev \circ \langle f, d\rangle\ R(\beta)\ MN \Rightarrow pev \circ \langle f, d\rangle\ R(\beta)\ M'N$,

that follows by ind. hyp. on β and (a).

(C.3): $\uparrow_{[\![1\to(\alpha\to\beta)]\!]}\ R(\alpha\to\beta)\ M$ because $pev\circ\langle\uparrow, d\rangle \cong \uparrow$, and $dR(\alpha)N$ implies, by induction hypothesis on β, $\uparrow R(\beta)\ MN$.

(C.4) $pev\circ\langle\sqcup_{n<\omega}f_n, d\rangle = \sqcup_{n<\omega}pev\circ\langle f_n, d\rangle$, but $\forall n.\ f_n\ R(\alpha\to\beta)\ M$ and $d\ R(\alpha)\ N$ implies $\forall n.\ pev\circ\langle f_n, d\rangle\ R(\beta)\ MN$. The thesis follows by (C.4) over $R(\beta)$. □

Theorem

If $\vdash(x_1: \alpha_1),\ldots,(x_n: \alpha_n) \supset M: \alpha$ and $d_i R(\alpha_i) C_i$, $i=1,..,n$ then

$[\![\Gamma \supset M: \alpha]\!] \circ \langle d_1,\ldots,d_n\rangle\ R(\alpha)\ [C_1/x_1,\ldots, C_n/x_n]M$.

Proof

By induction on the length of the typing judgment. We adopt the following abbreviations: $[\![\Gamma \supset M: \alpha]\!] \circ \langle d_1,\ldots,d_n\rangle \equiv [\![\Gamma \supset M: \alpha]\!] \circ \langle \mathbf{d}\rangle_n$, $[C_1/x_1,\ldots, C_n/x_n]M \equiv [\mathbf{C}/\mathbf{x}]_n M$.

(*) $[\![\Gamma \supset *: 1]\!] \circ \langle \mathbf{d}\rangle_n\ R(1)\ [\mathbf{C}/\mathbf{x}]_n *$, by definition of $R(1)$.

(asmp) $d_i R(\alpha_i) C_i$, by assumption.

($\to$.Intro) We have to show: $p\Lambda([\![\Gamma,x:\alpha \supset M:\beta]\!]) \circ \langle \mathbf{d}\rangle_n\ R(\alpha\to\beta)\ \lambda x:\alpha.[\mathbf{C}/\mathbf{x}]_n M$. The first condition that defines $R(\alpha\to\beta)$ follows by the fact that $\lambda x:\alpha.[\mathbf{C}/\mathbf{x}]_n M \downarrow$. For the second suppose $d\ R(\alpha)\ N$, $\vdash N \mapsto C$, and the application is defined, then by inductive hypothesis we have:

$[\![\Gamma,x:\alpha \supset M:\beta]\!] \circ \langle\langle \mathbf{d}\rangle_n, d\rangle\ R(\beta)\ [C/x, C_1/x_1,\ldots, C_n/x_n]M$. Observe:

(i) $pev \circ \langle p\Lambda([\![\Gamma,x:\alpha \supset M:\beta]\!]) \circ \langle \mathbf{d}\rangle_n, d\rangle = [\![\Gamma,x:\alpha \supset M:\beta]\!] \circ \langle\langle \mathbf{d}\rangle_n, d\rangle$.

(ii) $\vdash [C/x, C_1/x_1,\ldots, C_n/x_n]M \mapsto C' \Rightarrow \vdash (\lambda x:\alpha.[\mathbf{C}/\mathbf{x}]_n M)N \mapsto C'$

(iii) Hence by condition (C.2) follows:

$pev \circ \langle p\Lambda([\![\Gamma,x:\alpha \supset M:\beta]\!]) \circ \langle \mathbf{d}\rangle_n, d\rangle\ R(\beta)\ (\lambda x:\alpha.[\mathbf{C}/\mathbf{x}]_n M)N$.

($\to$.Elim) We have to show:

$pev\circ\langle [\![\Gamma \supset M: \alpha\to\beta]\!], [\![\Gamma\supset N:\alpha]\!]\rangle \circ \langle \mathbf{d}\rangle_n\ R(\beta)\ [\mathbf{C}/\mathbf{x}]_n(MN)$. By ind. hyp.

$[\![\Gamma \supset M: \alpha\to\beta]\!] \circ \langle \mathbf{d}\rangle_n\ R(\alpha\to\beta)\ [\mathbf{C}/\mathbf{x}]_n M$ and $[\![\Gamma\supset N:\alpha]\!] \circ \langle \mathbf{d}\rangle_n\ R(\alpha)\ [\mathbf{C}/\mathbf{x}]_n N$.

The result follows by the definition of $R(\alpha\to\beta)$.

(Y.Intro) We have to show: $\sqcup_{n<\omega} f(n) \circ \langle \mathbf{d}\rangle_n\ R(\alpha)\ Y[\mathbf{C}/\mathbf{x}]_n M$. We prove by induction that, for each n, $f(n) \circ \langle \mathbf{d}\rangle_n\ R(\alpha)\ Y[\mathbf{C}/\mathbf{x}]_n M$.

For the base case: $f(0) \circ \langle \mathbf{d}\rangle_n\ R(\alpha)\ Y[\mathbf{C}/\mathbf{x}]_n M$ follows by (C.3).

For the induction step observe: $pev\circ\langle g, \underline{id}\circ f(n)\rangle\ R(\alpha)\ M(\lambda x:1.Y[\mathbf{C}/\mathbf{x}]_n M)$ by ind. hyp. on $\Gamma\supset M:(1\to\alpha)\to\alpha$. Now use (C.2) to conclude:

$pev\circ\langle g, \underline{id}\circ f(n)\rangle\ R(\alpha)\ Y[\mathbf{C}/\mathbf{x}]_n M$. Hence by (C.4) we have the thesis. □

Corollary

(1) If $\vdash \varnothing \supset M: \alpha$ then $(M\Downarrow \Rightarrow M\downarrow)$.

(2) If $\vdash \Gamma \supset M: \alpha$, $\vdash \Gamma \supset N: \alpha$, and $[\![\Gamma \supset M: \alpha]\!] \leq [\![\Gamma \supset N: \alpha]\!]$ then in any context $C[\ ]$ such that $\vdash \varnothing \supset C[M]: \beta$, $\vdash \varnothing \supset C[N]: \beta$ we have: $C[M]\downarrow \Rightarrow C[N]\downarrow$.

Proof

(1) Apply the theorem in the case the context is empty.

(2) Prove by induction on the structure of a context C[ ] that:
if $[\![\Gamma \supset M: \alpha]\!] \leq [\![\Gamma \supset N: \alpha]\!]$, $\vdash \varnothing \supset C[M]:\beta$, and $\vdash \varnothing \supset C[N]: \beta$
then $[\![\varnothing \supset C[M]:\beta]\!] \leq [\![\varnothing \supset C[N]: \beta]\!]$.

Next observe: $C[M]\downarrow \Rightarrow_{(a)} C[M]\Downarrow \Rightarrow_{(b)} C[N]\Downarrow \Rightarrow_{(a)} C[N]\downarrow$.
where: (a) follows by (1), and (b) follows by $[\![\varnothing \supset C[M]:\beta]\!] \leq [\![\varnothing \supset C[N]: \beta]\!]$.

Analysis of the Proof

We wish to suggest where the definition of adequacy relation comes from. In organizing our proof we assume that we have arrived at the following straightforward formulation and that we try to proceed by induction on the length of the proof of typing.

Attempt 1. If $\vdash (x_1: \alpha_1),...,(x_n: \alpha_n) \supset M: \alpha$ and $d_i R_1(\alpha_i) C_i$, i=1,..,n then
$[\![\Gamma \supset M: \alpha]\!] \circ \langle d_1,...,d_n\rangle\ R_1(\alpha)\ [C_1/x_1,..., C_n/x_n]M$
where: $d\ R_1(\alpha)\ M$ iff $d\Downarrow \Rightarrow M\downarrow$.

This attempt already fails at the ($\rightarrow$.Elim) clause. Roughly suppose $pev \circ \langle f, d\rangle\Downarrow$, then we can conclude $f\Downarrow$ and $d\Downarrow$, so by induction hypothesis we can say $M \mapsto \lambda x{:}\alpha.M'$, and $N \mapsto C'$. But to show $MN\downarrow$ according to the corresponding evaluation rule we also need to prove $[C'/x]M'\downarrow$, and this seems outside our possibilities. Hence we are led to the idea of defining a new relation R_2 that satisfies: (a) $d\ R_2(\alpha)\ M \Rightarrow d\ R_1(\alpha)\ M$, and (b) $f\ R_2(\alpha\rightarrow\beta)\ M \wedge d\ R_2(\alpha)\ N \Rightarrow pev \circ \langle f, d\rangle\ R_2(\alpha)\ MN$. The inductive way to do this is to define an "exponent" over relations as shown in the previous section. Given that $R_2(\alpha)$ denotes such a "provisional adequacy relation" associated to the type α we can state our

Attempt 2. If $\vdash (x_1: \alpha_1),...,(x_n: \alpha_n) \supset M: \alpha$ and $d_i R_2(\alpha_i) C_i$, i=1,..,n then
$[\![\Gamma \supset M: \alpha]\!] \circ \langle d_1,...,d_n\rangle\ R_2(\alpha)\ [C_1/x_1,..., C_n/x_n]M$.

We now run into problems with the ($\rightarrow$.Intro) clause. Roughly we need the following closure property: $(f\ R_2(\beta)\ [C/x]M) \Rightarrow f\ R_2(\beta)\ (\lambda x{:}\alpha.M)C$. Hence it looks like "adequacy relations" should be closed under expansions that preserve convergence. Condition (C.2) strongly abstract from this empirical observation by requiring that an "adequacy relation" is invariant w.r.t. the relation $\approx$ we have previously described. Clearly this is a very desirable condition as: (i) it is semantically appealing, (ii) it refers to the evaluation relation rather to the single rules of the evaluation relation, (iii) it avoids completely the notion of "one-step reduction" whose introduction would lead us to a complex syntactic analysis. Henceforth we assume that our "provisional adequacy relations" satisfy (C.2). This attempt actually works nicely for the recursion free part of the calculus, and by

simple inspection of the previous proofs we have the following:

Fact. If $\vdash(x_1: \alpha_1),...,(x_n: \alpha_n) \supset M: \alpha$ and $d_i R_2(\alpha_i) C_i$, i=1,..,n, *where M, C_i do not include the Y combinator,* then

$$[\![\Gamma \supset M: \alpha]\!] \circ \langle d_1,...,d_n\rangle\, R_2(\alpha)\, [C_1/x_1,...,C_n/x_n]M\,.$$

New problems arise when we come to the (Y.Intro) clause. Since the fixed point is computed as the limit of an iteration, and the term is evaluated through an unfolding, computational induction seems the natural way to proceed. Of course this will work only if we make adequate hypothesis on the adequacy relations, namely existence of a least element (C.3) and directed completeness (C.4). Again we have to check that the "exponent" preserves this properties, but, as for (C.2), this is somehow folklore.

Exercise: Add the following condition to the definition of adequacy relation:

(C.5) $\quad f\, S\, M \Rightarrow f \le [\![\varnothing \supset M:\alpha]\!]$

and try to reprove the adequacy theorem.

4. Control Operators and CPS Translations

Most programming languages whose basic kernel is based on typed lambda calculus also include "control" operators such as exceptions or call-with-current-continuation (see for instance Scheme or ML). In this section it is shown how to *type* certain control operators and how to give them an adequate functional interpretation. As already hinted in previous examples the monad of continuations is a useful technical tool in order to approach these problems.

A Fragment of PCF with Control Operators

For the sake of simplicity we consider the following fragment of the language PCF presented in chpt 1.

$$\alpha ::= \text{num} \mid (\alpha\to\alpha) \qquad \Gamma ::= \varnothing \mid \Gamma, x:\alpha \qquad M ::= n \mid x \mid \lambda x{:}\alpha.M \mid MM \quad (n\in\omega)$$

The language of terms is enriched by two unary combinators C (for control) and A (for abort). Hence we have:

$$M ::= ... \mid CM \mid AM$$

As in the call by value lambda calculus we define a collection of canonical forms, these are the closed terms generated by the following grammar:

$$V ::= n \mid \lambda v{:}\alpha.M$$

In order to formalize the behaviour of C and A we introduce the notion of *evaluation context* E[ ]:

$$E[\,] ::= [\,] \mid E[\,]M \mid (\lambda v{:}\alpha.M)E[\,]$$

Note that an evaluation context is a context with exactly one hole which is not in the scope of a lambda abstraction. If we forget about type labels the one step reduction relation on terms is defined as follows:

(β) $E[(\lambda x.M)V] \rightarrow E[[V/x]M]$
(C) $E[CM] \rightarrow M(\lambda x.\, AE[x])$
(A) $E[AM] \rightarrow M$

We are now in a position to provide a syntactic intuition of what is a continuation for a given term and of what is special about a control operator. A redex Δ is defined as follows: $\Delta ::= (\lambda x.M)V \mid CM \mid AM$. Given a term $M \equiv E[\Delta]$, the *current continuation* is the abstraction of the evaluation context, that is $\lambda x.E[x]$. We will see later that there is at most one decomposition of a term into an evaluation context $E[\,]$ and a redex Δ. A *control operator* is a combinator which can manipulate directly the current continuation. In particular the operator A disregards the current continuation and starts the execution of its argument, while the operator C applies the argument to a variant of the current continuation.

Example: We illustrate by an example the role of control operators in functional programming. We want to write a function F: Tree(num)→num where Tree(num) is a given type of binary trees whose nodes are labelled by natural numbers. The function F has to return the sum of labels of the tree nodes but if it finds that a node has label 0, in this case it has to return zero in a *constant* number of steps of reduction. Intuitively the termination time has to be independent from the size of the current stack of recursive calls. There is a simple answer to this specification using the abort operator:

```
let     F(t) = F'(λx.Ax)t
where   F' = λk.Y(λf.λt'.   if empty(t') then 0
                            else  if num(t') = 0 then k0
                                  else num(t')+f(left(t'))+f(right(t'))
```

At the beginning of the computation one has:

$F(t) \rightarrow [\lambda x.Ax/k]F'$

If at some point the exceptional branch "if num(t') = 0 ..." is selected then the following computation is derived, in some evaluation context E[ ]:

$E[(\lambda x.Ax)0] \rightarrow E[A0] \rightarrow 0$

By applying the CPS translation described in the following it is possible to obtain a purely functional program with a similar behaviour. Note however that the design of this program is not obvious and its behaviour is probably more obscure than that of the program with control operators. On this basis one may advocate a direct study of control operators. Unfortunately a development of this point would take us too far away. See Felleisen&al.[87] for a syntactic analysis. □

Typing Control Operators

It is possible to type the operators C and A coherently with the reduction rules as follows. Let $\neg\alpha \equiv \alpha\rightarrow$num. Add to the standard rules (asmp), (I), (E), and the axiom for numerals the following two typing rules.

(C) $\Gamma \supset M{:}\neg\neg\alpha \Rightarrow \Gamma \supset (CM){:}\ \alpha$

(A) $\Gamma \supset M{:}$ num $\Rightarrow \Gamma\supset(AM){:}$ num

A *program* is a closed term of type num. The evaluation rules (β), (C), (A) apply to programs and are rewritten with all the type information as follows:

(β) $E[(\lambda x{:}\alpha.M)V] \rightarrow E[[V/x]M]$

(C) $E[CM] \rightarrow M(\lambda x{:}\alpha.\ AE[x])$ (where: $\varnothing \supset CM{:}\ \alpha$)

(A) $E[AM] \rightarrow M$

The rules above define a deterministic procedure to reduce a closed term. A subscript "C" will indicate we are considering the fragment of PCF described above extended with the control operators C and A and the relative typing and reduction rules.

Proposition *(Unique Decomposition)*

Suppose $\vdash_C \varnothing \supset M{:}\ \alpha$. Then either M is a canonical term or there is a unique evaluation context E[ ] and redex Δ such that $M \equiv E[\Delta]$.

Proof

By induction on M structure. The only interesting case is when $M \equiv M'M''$. Then M is not in canonical form and $\vdash_C \varnothing \supset M'{:}\ \beta\rightarrow \alpha$, $\vdash_C \varnothing \supset M''{:}\ \beta$, for some β.

- M' is canonical. Then $M' \equiv \lambda x{:}\alpha.M_1$. If M'' is canonical take $E[\ ] \equiv [\ ]$ and $\Delta \equiv (\lambda x{:}\alpha.M_1)M''$. Otherwise if M'' is not canonical then, by inductive hypothesis, there are $E_1[\ ]$, Δ_1 such that $M'' \equiv E_1[\Delta_1]$. Then take $E[\ ] \equiv M'E_1[\ ]$ and $\Delta \equiv \Delta_1$.

- M' is not canonical. Then, by inductive hypothesis, there are $E_1[\ ]$, Δ_1 such that $M' \equiv E_1[\Delta_1]$. Then take $E[\ ] \equiv E_1[\ ]M''$ and $\Delta \equiv \Delta_1$.

In all cases verify that there are no alternatives to the decomposition proposed. □

Proposition *(Subject Reduction for C and A)*

If $\vdash_C \varnothing \supset M{:}$ num and $M \rightarrow_C N$ then $\vdash_C \varnothing \supset N{:}$ num.

Proof

Suppose there are E[ ], Δ such that $M \equiv E[\Delta]$. There are three cases to consider according to the shape of the redex.

- $\Delta \equiv (\lambda x{:}\alpha.M)V$. This requires a simple form of the substitution lemma. Observe $\vdash_C x{:}\alpha \supset M{:}\ \beta$ and $\vdash_C \varnothing \supset V{:}\ \alpha$ implies $\vdash_C \varnothing \supset [V/x]M{:}\ \beta$.

- $\Delta \equiv CM$. Suppose $\vdash_C \varnothing \supset CM{:}\ \alpha$. Then $\vdash_C \varnothing \supset M{:}\ \neg\neg\alpha$ and $\vdash_C x{:}\alpha \supset E[x]{:}$ num. Hence $\vdash_C x{:}\alpha \supset AE[x]{:}$ num which implies $\vdash_C \varnothing \supset \lambda x{:}\alpha.AE[x]{:}\ \neg\alpha$ and finally $\vdash_C \varnothing \supset M(\lambda x{:}\alpha.AE[x]){:}$ num.

- $\Delta \equiv AM$. $\vdash_C \varnothing \supset AM$: num forces $\vdash_C \varnothing \supset M$: num. Also by definition of program $\vdash_C \varnothing \supset E[AM]$: num . □

The previous propositions show that the rules (β), (C), (A) when applied to a program define a deterministic evaluation strategy which preserves the well-typing.

Exercise Adef: Show that the operator C can simulate the operator A while being consistent with the typing given above. Hint: replace any occurrence of AM by $C(\lambda k{:}\neg num.M)$ where k is a fresh variable.

CPS Translation

Next we describe an interpretation of the λ_C calculus into the fragment without control operators. We begin with a translation on types. The interpretation of the arrow follows the monadic view where we take num as the type of results. From another point of view observe that replacing num with ⊥ one obtains a fragment of a translation from intuitionistic to classical logic (see Griffin[90], Murthy[91] for elaborations over this point).

$\underline{num} = num \qquad \underline{\alpha \to \beta} = \underline{\alpha} \to \neg\neg\underline{\beta}$

We associate to a term M a term $\underline{M}$ without control operators so that:

$\vdash_C x_1{:}\alpha_1,\ldots, x_n{:}\alpha_n \supset M{:}\alpha$ implies $\vdash x_1{:}\underline{\alpha_1},\ldots, x_n{:}\underline{\alpha_n} \supset \underline{M}{:} \neg\neg\underline{\alpha}$

The definition goes as follows (we omit types). This is known as Continuation Passing Style translation.

$\underline{x} = \lambda k.kx$ $\qquad$ $\underline{MN} = \lambda k.\underline{M}(\lambda m.\underline{N}(\lambda n.mnk))$

$\underline{n} = \lambda k.kn$ $\qquad$ $\underline{CM} = \lambda k.\underline{M}(\lambda m.m(\lambda z.\lambda d.kz)\lambda x.x)$

$\underline{\lambda x.M} = \lambda k.k(\lambda x.\underline{M})$ $\qquad$ $\underline{AM} = \lambda k.\underline{M}(\lambda x.x)$

Before giving the explicit typing of the translation we recall three basic combinators of the continuation monad.

$M{:}\alpha$ $\qquad$ $\eta(M) \triangleq \lambda k{:}\neg\alpha.\ kM : \neg\neg\alpha$

$M{:}\alpha \to \beta$ $\qquad$ $\neg\neg M \triangleq \lambda k{:}\neg\neg\alpha.\ \lambda h{:}\neg\beta.\ k(\lambda x{:}num.h(Mx))$

$M{:}\neg\neg\neg\neg\alpha$ $\qquad$ $\mu(M) \triangleq \lambda k{:}\neg\alpha.\ M(\lambda h{:}\neg\neg\alpha.\ hk){:}\neg\neg\alpha$

The explicitly typed CPS translation is:

$\underline{x} = \lambda k{:}\neg\underline{\alpha}.kx{:}\neg\neg\underline{\alpha}$

$\underline{n} = \lambda k{:}\neg num.kn{:}\neg\neg\underline{num}$

$\underline{\lambda x{:}\alpha.M} = \lambda k{:}\neg(\underline{\alpha}\to\underline{\beta}).\ k(\lambda x{:}\underline{\alpha}.\underline{M}){:}\neg\neg(\underline{\alpha}\to\underline{\beta})$

$\underline{MN} = \lambda k{:}\neg\underline{\beta}.\ \underline{M}(\lambda m{:}\ \underline{\alpha}\to\underline{\beta}.\ \underline{N}(\lambda n{:}\underline{\alpha}.\ mnk)){:}\neg\neg\underline{\beta}$

$\underline{CM} = \lambda k{:}\ \neg\underline{\alpha}.\underline{M}(\lambda m{:}\neg\neg\underline{\alpha}.m(\lambda z{:}\underline{\alpha}.\lambda d{:}\neg num.kz)(\lambda x{:}num.x))$

$\underline{AM} = \lambda k{:}\ \neg num.\underline{M}(\lambda x{:}\ num.x)$:

It is now a matter of verification to prove the following, where $\underline{\Gamma, x:\alpha} \equiv \underline{\Gamma}, x:\underline{\alpha}$.

Proposition *(typing CPS translation)*
Given the previous translation, if $\vdash_C \Gamma \supset M: \alpha$ then $\vdash \underline{\Gamma} \supset \underline{M}: \neg\neg\underline{\alpha}$.

Exercise CPS': There are many possible CPS translations. In particular verify that, consistently with the proposed typing, one can give the following translation of application: $\underline{MN} = \lambda k{:}\neg\underline{\beta}.\underline{N}(\lambda n{:}\ \underline{\alpha}.\underline{M}(\lambda m{:}\underline{\alpha\to\beta}.mnk))$.

The main problem is now to show that the CPS translation adequately represents the intended behavior of the control operators. The desired result reads as follows:

Suppose $\vdash_C \varnothing \supset M: \text{num}$. Then $M \to_C^* \underline{n}$ iff $\underline{M}\ \text{id} \to_\beta^* \underline{n}$.

The difficulty in proving this result consists in relating reductions of M and $\underline{M}$ id.

Example It is *not* the case that for M: num $M \to_C N$ implies $\underline{M}$ id $\to_\beta^* \underline{N}$ id. Consider for instance $(\lambda x.x)(Ap) \to_C p$. Note: $\underline{(\lambda x.x)(Ap)} \to_\beta^* \lambda k.p$, whereas $\underline{p} \equiv \lambda k.kp$.

An OptimizedTranslation

Given a term M a new translation $\langle M\rangle \equiv \lambda k.\ M: k$ is defined with the following relevant properties: (i) $\underline{M} \to_\beta^* \langle M\rangle$, (ii) if M: num and $M \to N$ then $M{:}id \to_\beta^* N{:}id$. We limit our attention to the fragment of the calulus *without* control operator. An extension of the results to the full calculus is possible but it would require a rather long detour (see Danvy&Filinski[92]).

The translation considered here, also known as colon translation, performs a more careful analysis of the term, the result is that a certain number of redexes can be statically reduced. Suppose: $U ::= \underline{n} \mid x \mid \lambda v{:}\alpha.M$ (in particular a canonical form is a U term). Inductively define the following translations on terms (a variant of the one in Plotkin[75]):

ψ-translation. Defined on U terms. Expected typing: if $U: \alpha$ then $\psi(U): \underline{\alpha}$.

$\psi(x) = x \qquad \psi(n) = n \qquad \psi(\lambda x{:}\alpha.M) = \lambda x{:}\underline{\alpha}.\underline{M}$

Exercise SUB: Any V, $[\psi(V)/x]\ \underline{M} \equiv \underline{[V/x]M}$

Semi-colon translation. Defined on a pair M: U. Expected typing: if $M{:}\alpha$ and $U{:}\ \neg\underline{\alpha}$ then M:U: num.

$U: k = \quad k\ \psi(U)$
$U_1U_2 : k = \quad \psi(U_1)\psi(U_2)k$
$U_1N : k = \quad N{:}\ \lambda n.\psi(U_1)nk$
$MN{:}\ k = \quad M{:}\ (\lambda m.\underline{N}(\lambda n.mnk))$

It is easily verified that this definition is well-founded. Let K be the least set of well typed terms (continuations) generated by the following grammar:

$k ::= \lambda x.x \mid \lambda n.\psi(U)nk \mid \lambda m.\underline{N}(\lambda n.mnk)$

It is convenient to associate a continuation $k^{E[\ ]} \in K$ to any evaluation context E[ ] as follows:

$k[\ ] \equiv \lambda x.x$

$k^{E[[\ ]N]} \equiv \lambda m.\underline{N}(\lambda n.mnk^{E[\ ]})$

$k^{E[U[\ ]]} \equiv \lambda n.\psi(U_1)nk^{E[\ ]}$

Lemma *(A)*

Suppose $\vdash \Gamma \supset M: \alpha$. Then (1) $\vdash \underline{\Gamma} \supset \langle M \rangle: \neg\neg\underline{\alpha}$ and

(2) If $k \in K$, and $k: \neg\underline{\alpha}$ then $\underline{M}k \rightarrow^+ M:k$.

Proof

By induction on M and case analysis of the semi-colon translation. □

Lemma *(B)*

Suppose $E[E'[(\lambda x.M)V]]$: num then

$E'[(\lambda x.M)V]:k^{E[\ ]} \rightarrow^+ E'[[V/x]M]:k^{E[\ ]}$.

Proof

By induction on E'[ ] structure. □

Theorem *(Adequacy CPS-translation)*

Suppose $\vdash_C \varnothing \supset M:$ num. Then $M \rightarrow^* \underline{n}$ iff $\underline{M}$ id $\rightarrow^* \underline{n}$.

Proof

($\Rightarrow$) Suppose $M \rightarrow^* \underline{n}$. By lemma A: $\underline{M}$ id $\rightarrow_\beta^*$ M: id. By iterated application of lemma B: M: id $\rightarrow_\beta^* \underline{n}$: id. By definition of the :-translation $\underline{n}$: id $\rightarrow_\beta \underline{n}$.

($\Leftarrow$) By strong normalization of the fragment of PCF considered here $\underline{M}$ id has to reduce to a normal form. Since $\underline{M}$ id is a closed term the normal form has to be a numeral, say n. Also the program M evaluates to a numeral m. By the first part of the proof $\underline{M}$ id $\rightarrow_\beta^* \underline{m}$. By confluence of β-reduction n=m. □

Exercise CBV-CBN: Given a program M show that in a call-by-value evaluation of the term M:id one never reduces in an evaluation context of the shape VE[ ]. Conclude that for this term call-by-name and call-by-value reduction coincide.

5. Monads of Powerdomains

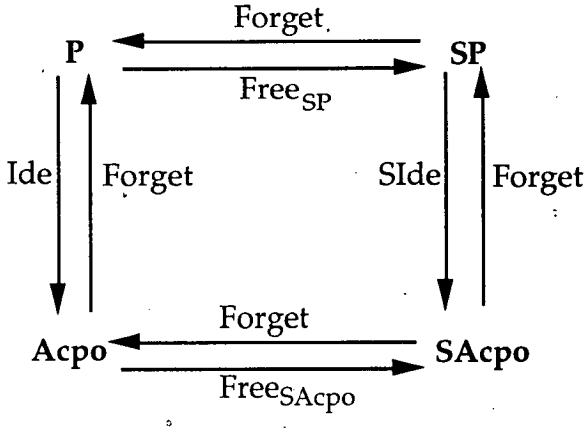
In this section we reconsider the monad of non-deterministic computations in the framework of domain theory. The need for this development clearly arises when we consider recursion together with a non-deterministic operator. Unfortunately in the context of domain theory there is not a unique candidate which can play the role of the collection of finite subsets in the category of sets. As

a matter of fact there are several possible constructions and their success might depend on the specific application one is considering. We will not investigate these applications here, our goal being just to give a synthetic mathematical introduction to three basic 'powerdomains'.

A rather abstract way to describe powerdomain constructions is to look at them as free constructions over certain semi-lattices.

- A *semi-lattice* is a set with a binary operation, say $*$, that is associative, commutative, and absorptive, i.e. $(x*y)*z = x*(y*z)$, $x*y = y*x$, $x*x = x$.
- We consider algebras over pre-orders ((algebraic) cpos), in this case we require that the carrier is a pre-order ((algebraic) cpo) and the operations are monotone (continuous).
- A pre-order with a monotone binary operation, say $(P, \leq, *)$ is a *join-semilattice* if it satisfies: $x \leq x*y$, and a *meet-semilattice* if it satisfies: $x*y \leq x$.

We are interested in the problem of showing that given an algebraic cpo there is a *freely generated* (join/meet) semi-lattice over the category of algebraic cpos. In view of the technique of ideal completion this problem can be actually decomposed in the problem of freely generating a semi-lattice over the category of preorders and then completing it to a semi-lattice over the category of algebraic cpos. Here is the basic schema for semi-lattices, where:



- **P** is the category of pre-orders and monotone maps.
- **SP** is the category of pre-ordered semilattices and monotone homo-morphisms.
- **Acpo** is the category of algebraic cpos and Scott continuous maps.
- **SAcpo** is the category of algebraic cpos semilattices and continuous homo-morphisms.
- Forget are the appropriate functors that forget some of the structure, in the diagram. They are intended as *right adjoints* (if there is a left adjoint !).
- Ide is the ideal completion from preorders to algebraic cpos.
- SIde is the ideal completion on the carrier of the algebra and the extension of the monotone operations to continuous ones.
- Free$_{SAcpo}$ is the functor we want to show to exist as left adjoint to Forget.

• In order to show the existence of $\text{Free}_{\text{SAcpo}}$ it is actually enough to exhibit a functor Free_{SP} left adjoint to Forget that takes a pre-order and freely generates the semi-lattice over the category P. Then we define:

$$\text{Free}_{\text{SAcpo}}(D) \triangleq \text{SIde}(\text{Free}_{\text{SP}}(D_o)), \qquad \text{Free}_{\text{SAcpo}}(f) \triangleq \text{SIde}(\text{Free}_{\text{SP}}(f \mid D_o)).$$

We are now ready to prove the main result:

Theorem

The functor Forget: **SP**→**P** has a left adjoint Free_{SP}: **P**→**SP** that is defined as:

$$\text{Free}_{\text{SP}}(P) \triangleq (P_{\text{fin}}(P), \leq_c, \cup), \quad \text{Free}_{\text{SP}}(f)(X) \triangleq f(X),$$

where the so-called *convex* pre-order is defined as:

$$X \leq_c Y \Leftrightarrow \forall x \in X.\, \exists y \in Y.\, (x \leq y) \wedge \forall y \in Y.\, \exists x \in X.\, (x \leq y)$$

and the semi-lattice operation is the set-theoretic union.

Proof

Define the natural transformation $\tau_{P,S}$: $\mathbf{P}[P, \text{Forget}(S)] \rightarrow \mathbf{SP}[\text{Free}_{\text{SP}}(P), S]$ as:

$$\tau_{P,S}(f)(X) \triangleq f(x_1)...f(x_n),$$

where $X = \{x_1,...,x_n\} \in P_{\text{fin}}(P)$ and the operation in S is simply represented by juxtaposition. The inverse is defined as: $\tau_{P,S}^{-1}(h)(p) \triangleq h(\{p\})$.

We have to verify that these are morphisms in the respective categories.

• $\tau_{P,S}(f)$ is monotone. Suppose $\{x_1,...,x_n\} = X \leq_c Y = \{y_1,...,y_m\}$. By the definition of the convex pre-order we can find two multisets $X'=\{w_1,...,w_l\}$ and $Y'=\{z_1,...,z_l\}$ in which occur the same elements, respectively, as in X and Y and such that $w_i \leq z_i$, $i=1,...,l$. By monotonicity of f and the * operation in S we have: $f(w_1)...f(w_l) \leq_S f(z_1)...f(z_l)$, and by absorption and the assumption on X', Y': $\tau_{P,S}(f)(X) = f(w_1)...f(w_l)$, $\tau_{P,S}(f)(Y) = f(z_1)...f(z_l)$.

• $\tau_{P,S}(f)$ is an S-morphism. Immediate by associativity and absorption.

We leave to the reader the verification that $\tau_{P,S}^{-1}$ is well defined as well as the check of the naturality of τ.

Let JSP the category of join semi-lattices and MSP the category of meet semi-lattices. The proofs of the following theorems have a similar pattern.

Theorem

The functor Forget: **JSP**→**P** has a left adjoint Free_{JSP}: **P**→**JSP** that is defined as:

$$\text{Free}_{\text{JSP}}(P) \triangleq (P_{\text{fin}}(P), \leq_l, \cup), \quad \text{Free}_{\text{JSP}}(f)(X) \triangleq f(X),$$

where the so-called *lower* pre-order is defined as:

$$X \leq_l Y \Leftrightarrow \forall x \in X.\, \exists y \in Y.\, (x \leq y)$$

and the join semi-lattice operation is the set-theoretic union.

Theorem

The functor Forget: **MSP**→**P** has a left adjoint $Free_{MSP}$: **P**→**MSP** defined as:

$Free_{MSP}(P) \triangleq (P_{fin}(P), \leq_u, \cup)$, $Free_{MSP}(f)(X) \triangleq f(X)$,

where the so-called *upper* pre-order is defined as:

$X \leq_u Y \Leftrightarrow \forall y \in Y.\, \exists x \in X.\, (x \leq y)$

and the meet semi-lattice operation is the set-theoretic union.

Exercise: Show that the category of Scott domains is closed under the lower and upper powerdomains constructions, but not the convex one. Show that the category of bifinite domains is closed under all the three powerdomains constructions.

Note: Observe the unfortunate combination of the terminologies for semi-lattices and pre-orders: the *lower* pre-order occurs with *join* semi-lattices, and the *upper* pre-order occurs with *meet* semi-lattices.

The Powerdomain Monad

Each adjunction is going to induce a monad over Acpo in the standard way. For example in the case of semilattices over preorders we have a monad (T, η, ε) defined as:

$T \triangleq Forget \circ Free_{SP} : P \to P$

$\eta_D: D \to TD$, $\eta_D(d) \triangleq \{d\}$

$\mu_D: T^2D \to TD$, $\mu_D(Z) \triangleq \cup Z$.

Clearly there are many more possibilities to build free algebras over algebraic cpos. More generally there might be powerdomains construction that are not describable as free constructions. A general problem with powerdomains is that their description is not "intrinsic" to the category we are considering (at least not in the approach presented here) but is rather a construction that we attach over the category of domains. Such constructions usually find their justification in some specific application to the semantics of non-determinism or relational data-bases.

Note *(on parallelism and non-determinism)*

Consider the following variant of the imperative language introduced in 1.1:

L_P: s::= a | dummy | s;s | s || s

The intuitive semantics of $s_1 \| s_2$ is that of a parallel execution of the state transformations performed by s_1 and by s_2 . Since s_1 and s_2 share the same store different orders of execution might generate different final results, as is clear, for instance, in the program 'x:=1; x:=0 || x:=1', which upon termination can associate to x either 0 or 1. A fundamental assumption is that modifications of the store can be serialized. So the program 'x:=0 || x:=1' will terminate with x having value 0 or 1, and nothing else. Also, because of the possibility of interleaving the execution of

various parallel statements it may happen that programs which were equivalent in a sequential framework need now distinct interpretations, e.g. 'x:=0; x:=x+1', and 'x:=1'.

To summarize the semantics of a statement s should now have the following type: $[\![s]\!]$: $(S \to (P_{fin}(S \to S)^*)$, i.e. a function which takes a state and returns a collection of finite strings of state transformations. We leave as an exercise the task of interpreting the statements in such a set (hint: this requires defining an operator which takes two strings and returns the collection of strings resulting from the possible shuffles). As expected in the presence of divergent programs things are a bit more complicated and what is usually needed is a recursively defined domain of 'resumptions' (see, e.g., Plotkin[83]) which involves the powerdomain monads studied above. □

Synchronization Trees and Powerdomains

As a another example of the application of powerdomains to the semantics of parallelism we present a method for building a domain of 'synchronization trees' associated to a given 'labelled transition system'.

Labelled Transition Systems

In first approximation the semantics of programs considered so far associates to every input a set of output values. For instance the empty set if the computation diverges, a finite set if the computation is non-deterministic but finitely branching, etcetera.

System applications often require the design of programs which have to interact with their environment (e.g. other programs, physical devices...). In this case the specification of a program as an input-output relation is *not* adequate (cf. previous example).

In order to specify the ability of a program to perform a certain action it is useful to introduce the simple notion of labelled transition system.

Definition *(Labelled Transition System)*

A triple of sets (Pr, Act, $\mapsto$) is a labelled transition system (lts) if $\mapsto \subseteq Pr \times Act \times Pr$.

We target our notation to a process calculus to be introduced next: Pr stands for the collection of processes and Act for the collection of actions. We write $p \mapsto^{\alpha} q$ for $(p, \alpha, q) \in \mapsto$, to be read as "p makes and an action α and becomes q". A lts is said to be *image finite* if: $\forall p \in Pr.\ \forall \alpha \in Act.\ \{p' \mid p \mapsto^{\alpha} p'\}$ is finite. An image finite lts can be represented as a function $\mapsto : Pr \times Act \to P_{fin}(Pr)$.

Example *(CCS)*

As a typical example of lts we consider Milner's Calculus of Communicating Systems (CCS). Let L be a countable collection of labels. Each label $l \in L$ has a complement l' which belongs to $L' \triangleq \{l' \mid l \in L\}$. The symbol " ' " can be understood

as a special marker that one adds to an element of L. The marker is chosen so that L and L' are disjoint. We denote with a, b,... generic elements in $L \cup L'$. The complement operation is extended to L' by making it involutive, that is: $a''=a$. Finally we define the collection of actions: $Act \triangleq L \cup L' \cup \{\tau\}$, where $\tau \notin L \cup L'$. We denote with α, β,... generic elements in Act.

Intuition. a, a' may be understood as input/output synchronization operations on a channel. The calculus models a protocol of communication in which sender and receiver have to synchronize in order to communicate on a channel. The action τ is an "internal action" in the sense that a process may perform it without cooperation of the environment. We define next a calculus of processes:

Process variables. X, Y, Z,...
Process expressions. $E ::= nil \mid X \mid \alpha.E \mid E+E \mid E \mid E \mid E\backslash a \mid E[a/b] \mid fix\, X.E$

A process is a *closed* process expression, i.e. all process variables are in the scope of a fix operator. We denote processes with P, Q... these are the objects to which an operational semantics will be assigned. The intuitive operational behaviour of the process operators is as follows: nil is the terminated process which can perform no action. $\alpha.E$ is the *prefixing* of α to E, that is $\alpha.E$ performs the action α and becomes E. E+E' is the *non-deterministic choice* (sum) between the execution of E and that of E'. This choice may depend on a "global" condition. E | E' is the *parallel composition* of E and E'. $E\backslash a$ is the process E where the channel a has become private to E. This operation is called *restriction*. E[a/b] is the process E where each action concerning the channel b is visible by the environment as a corresponding action on a. This operation is called *renaming*. Eventually fix is the fix-point operator with the usual unfolding operational semantics.

Labelled Transition System for CCS. We define next a lts for process expressions. The definition proceeds non-deterministically by analysis of the process expression structure. There are also rules (+r), (| r) symmetric to (+l), (| l). Finally let $\alpha[a/b] \triangleq$ if $\alpha \notin \{b, b'\}$ then α else (if $\alpha = b$ then a else a').

(.) $\Rightarrow \alpha.E \mapsto^{\alpha} E$

(+l) $E_1 \mapsto^{\alpha} E_1' \Rightarrow E_1+E_2 \mapsto^{\alpha} E_1'$

(| l) $E_1 \mapsto^{\alpha} E_1' \Rightarrow E_1 \mid E_2 \mapsto^{\alpha} E_1' \mid E_2$

(| τ) $E_1 \mapsto^{a} E_1'$, $E_2 \mapsto^{a'} E_2' \Rightarrow E_1 \mid E_2 \mapsto^{\tau} E_1' \mid E_2'$

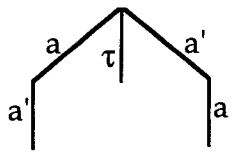
(\) $E \mapsto^{\alpha} E'$ $\alpha \notin \{a, a'\} \Rightarrow E\backslash a \mapsto^{\alpha} E'\backslash a$

([]) $E \mapsto^{\alpha} E' \Rightarrow E[a/b] \mapsto^{\alpha[a/b]} E'[a/b]$

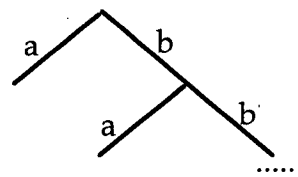
(fix) $[fix\, X.E/X]E \mapsto^{\alpha} E' \Rightarrow fix\, X.E \mapsto^{\alpha} E'$

Towards Synchronization Trees. Given a process P one may repeatedly apply the derivation rules above and build a possibly infinite tree whose edges are labelled by actions. For instance consider the following examples where nil processes are represented by the empty tree:

a.nil | a'.nil



fixX.a.nil+b.X



Exercises CCSTrees: (1) Show that all processes without a fix operator generate a finite tree. (2) Consider the process P ≡ fix X. ((X.nil+a.nil) | b.nil). Verify: $P \mapsto^a$ nil | b.nil | ... | b.nil, for an arbitrary number of b.nil's. Conclude CCS lts is not image finite.

The graphical representation is still too concrete to provide a reasonable semantics, even for finite CCS processes built out of prefixing and sum. In the first place the sum should be commutative and in the second place two identical subtrees with the same root should collapse into one. In other words the sum should form a *semi-lattice* with nil as identity. For processes generating a finite tree it is possible to build a canonical set-theoretic representation. Define inductively:

$$ST_0 = \emptyset \qquad ST_{n+1} = P_{fin}(Act \times ST_n) \qquad ST_\omega = \cup\{ST_n \mid n<\omega\}$$

If P generates a finite tree then let $[\![P]\!] = \{(a,[\![P']\!]) \mid P \mapsto^\alpha P'\}$. Going back to the previous example one can compute: $[\![a.nil \mid a'.nil]\!] = \{(a,\{(a', \emptyset)\}), (\tau, \emptyset), (a',\{(a, \emptyset)\})\}$.

Exercise: Verify that the previous interpretation is well-defined for processes generating a finite tree and that it satisfies the semi-lattice equations.

There are serious difficulties in extending this naive set-theoretic interpretation to infinite processes. For instance one should have:

$[\![fix\ X.a.X]\!] = \{(a, \{(a, \{(a,...$

This seems to ask for the construction of a set A such that $A=\{(a, A)\}$. Assuming the standard representation of an ordered pair (x,y) as {x, {x,y}} one notes that this set is not well-founded w.r.t. the "belongs to" relation as: $A \in \{a, A\} \in A$. This contradicts the foundation axiom which is traditionally added to, say, Zermelo-Fraenkel set-theory (see e.g. Jech[78]). On the other hand it is possible to remove the foundation axiom and develop a non-standard set-theory with an *anti-foundation* axiom which assumes the existence of sets like A (see in particular Aczel[88] for the development of the connections with process calculi). In the following we will take a different approach which pursues the construction of a structure of "synchronization trees" by relying in particular on the convex

powerdomain. First however it is useful to introduce a popular notion of equivalence on lts.

Bisimulation. Let (Pr, Act, $\mapsto$) be a given lts. We introduce an equivalence relation over Pr that can be characterized as the greatest element of a collection of relations known as bisimulations or, equivalently, as the greatest fix-point of a certain monotone operator defined on the powerset of binary relations on Pr.

Definition *($\mathcal{F}$-operator)*
Let (Pr, Act, $\mapsto$) be a given lts. Define $\mathcal{F}$: P(Pr×Pr)→P(Pr×Pr) as:

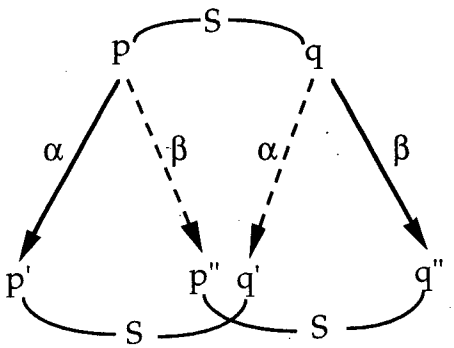
$$\mathcal{F}(X) \triangleq \{(p, q) \mid \forall p'.p \mapsto^{\alpha} p' \Rightarrow \exists q'.(q \mapsto^{\alpha} q' \wedge (p', q') \in X) \text{ and symmetrically } \forall q'.q \mapsto^{\alpha} q' \Rightarrow \exists p'.(p \mapsto^{\alpha} p' \wedge (p', q') \in X \}$$

Definition *(iterates of the $\mathcal{F}$-operator)*

$$\mathcal{F}^0 = \mathrm{Pr}\times\mathrm{Pr} \qquad \mathcal{F}^{\kappa+1} = \mathcal{F}(\mathcal{F}^{\kappa}) \qquad \mathcal{F}^{\lambda} = \bigcap \{\mathcal{F}^{\kappa} \mid \kappa<\lambda\} \text{ for } \lambda \text{ limit ordinal.}$$

Exercise $\mathcal{F}$: Show that $\mathcal{F}$ is a monotone operator over P(Pr×Pr) with a least and a greatest fix-point (gfp). In particular observe that $\mathrm{gfp}(\mathcal{F}) = \bigcap \{\mathcal{F}^{\kappa} \mid \kappa<\mu\}$, for some ordinal μ. If Pr is finite then one can take $\mu=\omega$. A bit harder: if the lts is finitely branching then one can take $\mu=\omega$.

Definition *(Bisimulation)*
Let (Pr, Act, $\mapsto$) be a given lts. A binary relation $S \subseteq \mathrm{Pr}\times\mathrm{Pr}$ is a bisimulation if $S \subseteq \mathcal{F}(S)$. The following diagram illustrates the properties of a bisimulation (plain and dashed arrows correspond to a universal and existential quantifier, respectively).



Exercise BIS: The empty and identity relations are bisimulations. Bisimulations are closed under inverse, composition and arbitrary unions. There is a greatest bisimulation. Bisimulation are not closed under (finite) intersection.

Proposition *(characterization greatest bisimulation)*
Let $\sim \triangleq \bigcup \{S \mid S \text{ bisimulation}\}$ be the greatest bisimulation. Then
$\sim = \mathrm{gfp}(\mathcal{F}) = \bigcap \{\mathcal{F}^{\kappa} \mid \kappa<\mu\}$, for some ordinal μ.

Exercise BIS-EQ: (1) Let Pr/~ be the collection of CCS processes quotient the greatest bisimulation. Show that Pr/~ induces a semi-lattice with operations + and nil. Analogously derive a commutative monoid from | and nil. (2) Show that ~ is a congruence for prefixing, sum, parallel composition, restriction and renaming.

Bisimulation and CCS Semantics. The previous exercise suggests that bisimulation equivalence captures many reasonable equivalences of processes. However as stated it is still unsatisfying in at least two respects:

(1) An internal action τ and an input-output action on a channel are treated in the same way. This implies that for instance the process $\tau.\tau.a.nil$ is not bisimilar to the process $\tau.a.nil$. This problem can be fixed by considering a modified lts defined as follows:

$$P \Rightarrow^a P' \text{ iff } P\ (\mapsto^\tau)^* \mapsto^a (\mapsto^\tau)^*\ P' \qquad P \Rightarrow^\tau P' \text{ iff } P\ (\mapsto^\tau)^*\ P'$$

The bisimulation built on top of this modified lts is known as *weak* bisimulation (the properties of this equivalence w.r.t. CCS operators are described in Milner[90]).

(2) Bisimulation is computed as a greatest fix-point. This has surprising effects on the semantics of recursive definitions. One proves for instance:

$$P \equiv \text{fix } X.\ a.nil+X \sim a.nil.$$

This equivalence seems to be incompatible with any interpretation of fix as a least-fix-point operator over some domain as one would derive:

$$[\![\text{fix } X.\ a.nil+X]\!] = \sqcup_{n<\omega} F^n(\perp) = [\![a.nil]\!]+\perp, \text{ where } F(A) = [\![a.nil]\!]+A$$

and one expects: $[\![a.nil]\!]+\perp \neq [\![a.nil]\!]$. The notion of bisimulation considered so far does not deal with the possibility that a process diverges. It is possible to enrich the notion of lts with a unary relation predicating the divergence of a process, and moreover one can define a notion of *partial* bisimulation over it (see, e.g., Abramsky[91]). This treatment of the operational semantics allows to get closer to a denotational interpretation of CCS as developed next.

A domain equation for bisimulation. The problem is to build a domain which extends to the infinite case our intuition about synchronization trees. We suppose that Act is a given countable collection of actions. We are interested in the "initial solution" of the following domain equation (see chapter 5 for a general theory of the solution of these equations).

$$D \cong P[(Act \times D)_\perp] \oplus (1)_\perp \qquad (*)$$

where: P[ ] is the convex powerdomain, $(\_)_\perp$ is the lifting, $\oplus$ is the coalesced sum, and 1 is the one point cpo. An element in D falls in one of the following three categories: (i) $\perp$ which represents the always diverging process, (ii) ø the non-bottom element of $(1)_\perp$ which denotes the terminated process nil, (iii) a "set"

including a collection of pairs $\{(a_i, d_i)\}_{i\in I}$ and possibly the bottom element which denotes the process which can perform the action a_i and become d_i or possibly diverge.

Concrete representation convex powerdomain. It is convenient to have a concrete representation of the elements of the convex powerdomain.

Definition *(*-operator)*
Let D be an algebraic cpo. For any $X\subseteq D$ define
$X^* = \{y \mid (\exists x\in X.x\leq y) \wedge (\forall d\in D_o.d\leq y \Rightarrow \exists x\in X.d\leq x)\}$

Exercise Cl*: Show that: (1) $X\subseteq X^*$, (2) $X^{**}=X^*$.

$P[D] = (\{X\subseteq D \mid X\neq\emptyset \wedge X=X^*\}, \leq_c)$
$P[f](X) = \{f(x) \mid x\in X\}^*$, given $f: D\to E$
$\eta \equiv \{|\ |\} : D\to P[D]$, $\{|d|\} = \{d\}^*$
$\mu \equiv \bigcup : P[P[D]]\to P[D]$, $\bigcup F = (\bigcup F)^*$
$\cup : P[D]^2\to P[D]$, $X\cup Y = (X\cup Y)^*$

All these operations can be extended to the convex powerdomain with an empty set adjoined: $P_\emptyset[D] \triangleq P[D] \oplus (1)_\perp$.

The *compact elements* of the domain D solving equation (*) are inductively defined as follows: (i) $\emptyset\in D_o$, (ii) $\{\perp\}\in D_o$, (iii) if $a\in Act$, $d\in D_o$ then $\{|(a,d)|\}\in D_o$, and (iv) if $d\in D_o$ and $d'\in D_o$ then $d\cup d'\in D_o$.

Interpretation. Nil, prefixing, sum and fix-point combinator are interpreted directly using the operations suggested by the construction of D.

$nil^D \triangleq \emptyset$
$a._-^D \triangleq \lambda d.\{|(a,d)|\}$
$+^D \triangleq \cup$
$fix^D \triangleq Fix$

The interpretation of restriction, relabelling and parallel composition is less direct. One has to visit the argument(s) and incrementally generate a result.

$_\backslash a^D \triangleq Fix(\lambda\Phi.\ \bigcup \circ P_\emptyset[g_a\Phi])$, where $g_a: (D\to D)\to((Act\times D)_\perp\to D)$,
$g_a\Phi\perp = \perp$
$g_a\Phi(\alpha, d) =$ if $a\neq\alpha$ then $(\alpha, \Phi\, d)$ else $\emptyset$

$_[a/b]^D \triangleq Fix(\lambda\Phi.\ P_\emptyset[g_{a/b}\Phi])$, where $g_{a/b}: (D\to D)\to((Act\times D)_\perp\to(Act\times D)_\perp)$,
$g_{a/b}\Phi\perp = \perp$
$g_{a/b}\Phi(\alpha, d) = (\alpha[a/b], \Phi\, d)$

$|^D \quad \triangleq \mathrm{Fix}(\lambda\Phi.\, f\Phi)^+$, where $f\colon (D^2 \to D) \to ((\mathrm{Act}\times D)_\perp{}^2 \to D)$

$f\Phi(x, \perp) = f\Phi(\perp, x) = \perp$

$f\Phi((\alpha,d), (\beta,d')) = \{|\ (\alpha,\Phi(d, (\beta,d'))\ |\} +^D \{|\ (\beta,\Phi(d', (\alpha,d'))\ |\}$
$+^D \{|\ (\tau,\Phi(d, d'))\ |\}$

the last addenda only if $\alpha=a$, $\beta = a'$.

We refer to Abramsky[91] for a comparison with the operational semantics. In general it is possible to prove a full abstraction theorem of this interpretation of finite terms w.r.t. partial bisimulation. The match is not quite so nice for infinite terms.

5. Domain Equations, Universal Domains, and Operator Representation

Contents: 1. Domain Equations, 2. Universal Domains, 3. Operator Representation.

Given a category of domains **C** one builds the related category $\mathbf{C}^{ep}$ that has the same objects as **C** and embedding-projection pairs as morphisms. It turns out that this is a suitable framework for the *solution of domain equations* and the construction of a *universal homogeneous object*. The latter is a domain in which every other domain (not exceeding a certain size) can be embedded. Once a universal object U is built, it is possible to *represent* the collection of domains as the domain of finitary projections over U, say FP(U), and functors as continuous transformations over FP(U). In this way, one obtains a poset-theoretic framework for the solution of domain equations that is more manageable than the general categorical one.

1. Domain Equations

One of the earliest problems in denotational semantics was that of building a model of the type-free $\lambda\beta\eta$-calculus. This boils down to the problem of finding a non-trivial domain D isomorphic to its functional space $D \to D$. Following Wand[79] and Smyth&Plotkin[82] we present a generalization of the technique proposed in Scott[72] for the solution of domain equations.

Fixed Points of Covariant Functors

In this section we introduce the notions of algebroidal category and of category of embedding-projection pairs. These notions can be found, e.g., in Banaschewski& Herrlich[76] and Smyth&Plotkin[82]. Recall that an ω-chain is a sequence $\{B_n, f_n\}_{n\in\omega}$ such that $f_n: B_n \to B_{n+1}$ for all n. It is convenient to write $f_{n,m} \triangleq f_{m-1} \circ \ldots \circ f_n$ for $m>n$.

The general categorical definition of colimit specializes to ω-chains as follows. A cocone $\{B, g_n\}_{n\in\omega}$ of the ω-chain $\{B_n, f_n\}_{n\in\omega}$ is given by an object B, and a sequence $\{g_n: B_n \to B\}_{n\in\omega}$ satisfying $g_{n+1} \circ f_n = g_n$ for all n. A cocone $\{B, g_n\}_{n\in\omega}$ is a colimit if it is an initial object in the category of cocones, that is if for any other cocone $\{C, h_n\}_{n\in\omega}$ there exists a unique arrow $k: B \to C$ such that $k \circ g_n = h_n$ for all n.

Let T: **K**→**K** be an endo-functor. We outline some rather general results that guarantee the existence of an "initial" solution for the equation TX ≅ X. It will be shown in the next section that these results can be usefully applied to the solution of domain equations.

Definition *(T-algebras)*

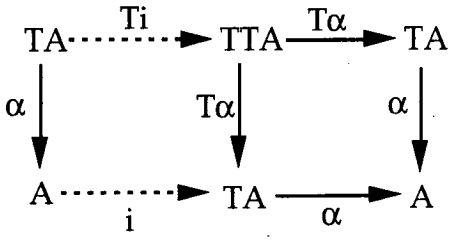
Let T: **K**→**K** be an endo-functor. A T-algebra is a map α: TA→A. T-algebras form a category. If α: TA→A and β: TB→B are T-algebras then a morphism from α to β is a map f: A→B such that: f∘α = β∘Tf.

Note *(on initial T-algebras)*

It is clear that any isomorphism i: TA→A gives rise to a T-algebra. Now we show that an initial T-algebra is always an isomorphism. Let α: TA→A be initial. Then Tα: TTA→TA is also a T-algebra and by initiality there is i: A→TA such that:

$$i \circ \alpha = T\alpha \circ Ti = T(\alpha \circ i) \quad (*).$$

Observe Tα is a morphism (of T-algebras) from Tα to α. By composition and initiality we get: α ∘ i = id. By (*) we get T(α ∘ i) = T(id) = id = i ∘ α. So i is the inverse of α. Here is the diagram:



The following proposition will appear natural if one thinks of categories as cpos and of functors as continuous transformations.

Proposition *(Initial Solution)*

Let **C** be a category with initial object and ω-colimits and T: **C**→**C** be a functor that preserves ω-colimits. Then there is an initial T-algebra.

Proof

Let O be the initial object. Consider the uniquely determined map z: O→TO. By iterating T on this diagram we get an ω-diagram: $D \equiv \{T^iO, T^iz\}_{i\geq 0}$. By assumption there is an ω-colimit of D, $C \equiv \{A, f^i\}_{i\geq 0}$, satisfying $f^i = f^{i+1} \circ T^iz$.

Now consider $TC \equiv \{TA, Tf^i\}_{i\geq 0}$. By assumption TC is an ω-colimit of: $TD \equiv \{TT^iO, TT^iz\}_{i\geq 0}$. Since we can restrict C to a cocone of TD we have determined a unique morphism h: TC→C. As for the inverse observe that TC can be extended to a cocone of D, as by initiality the first arrow of a cocone of C is fixed and the commutativity of the first triangle must hold. In this way we obtain a morphism k:C→TC that is the inverse of h.

We want to prove moreover that the T-algebra h: TA→A built in the solution of the equation TX ≅ X is initial. This goes in three steps:

(I) Observe that any T-algebra, β: TB→B, gives rise to a cocone $\{B, g^i_B\}_{i\geq 0}$ where:

$g^i_B \triangleq \beta \circ T\beta \circ TT\beta \circ \ldots \circ T^{i-1}\beta \circ T^i z_B$, with: $i\geq 0$, z_B: O→B.

(For the heuristics, think of the way you would prove $\cup f^n(\perp)\leq z$ if $f(z)\leq z$ in **Cpo**.)

Next we have to relate morphisms in Cocone($\{T^iO, T^iz\}_{i\geq 0}$) and morphisms of T-algebras. Observe:

(II) Given h: TA→A, β: TB→B the uniquely determined morphism l: A→B on the induced cocones is also a morphism of T-algebras. Observe that β also determines another cocone $\{TB, h^i_B\}_{i\geq 0}$ where: $h^i_B \triangleq T\beta \circ TT\beta \circ \ldots \circ T^i\beta \circ T^i z_{TB}$, with: $i\geq 0$, z_{TB} : O→TB, making β into a cocone morphism. Next remark that $l\circ h$ and $\beta\circ Tl$ are two cocone morphisms from TA to B, and since TA is a colimit, they are equal. Hence l is a T-algebra morphism.

(III) Any morphism of T-algebras u: $\alpha\rightarrow\beta$ induces a morphism between the related cocones, as defined in (I). □

In the practice of the solution of domain equations we may wish to start the construction of the ω-diagram with some morphism z: X→TX, where X is not necessarily an initial object. In the poset case this corresponds to looking for the least fixed point of a function, f: D→D, that is bigger than a given point d such that $d\leq f(d)$. If D is an ω-dcpo, and f is ω-continuous then the we can compute the solution as $\cup_{n<\omega} f^n(d)$. This is the least element of the set: $\{e\in D \mid f(e)\leq e \wedge d\leq e\}$.

We now want to provide a categorical generalization of this fact. Suppose that the category **C** and the functor F satisfy the condition in Prop. *(Initial Solution)*. Given the map z: X→TX we can build an ω-diagram $D \equiv \{T^iX, T^iz\}_{i\geq 0}$. Using the hypotheses we can build the colimit $\{A, f^i\}_{i\geq 0}$, and an isomorphism h: TA→A. The problem is now to determine in which framework h is initial. In first approximation it is natural to consider T-algebras β: TB→B *together* with a map z_B: X→B (as B has to be "bigger" than X). If we now try to mimic step (I) in the previous proof, that builds a cocone out of a T-algebra, we see that we need the following property: $z_B = \beta \circ Tz_B \circ z$. Generalizing step (II) is routine, but step (III) presents a new difficulty. It appears that a T-algebra morphism l: $\beta\rightarrow\gamma$, where β: TB→B, and γ:TC→C, should also satisfy $l \circ z_B = z_C$. The following categorical formalization shows that this is just an instance of the problem we have already solved, but w.r.t. to a related category $\mathbf{C}\uparrow X$, and a related functor T_z.

Given a category **C** and an object $X\in\mathbf{C}$, we define a new category $\mathbf{C}\uparrow X$ as:

$\mathbf{C}\uparrow X = \{f: X\rightarrow B \mid B\in\mathbf{C}\}$, $\mathbf{C}\uparrow X[f, g] = \{h \mid h\circ f = g\}$

Also given a functor T: **C**→**C**, and a morphism z: X→TX define a new functor:

T_z: $\mathbf{C}\uparrow X\rightarrow\mathbf{C}\uparrow X$, $T_z(f) = Th\circ z$ $T_z(h) = Th$

Proposition *(Relative Initial Solution)*
Let **C** be a category with initial object and ω-colimits and T: **C**→**C** be a functor that preserves ω-colimits. The category **C**↑X has initial object and ω-colimits, moreover given a morphism z: X→TX, the functor T_z preserves ω-colimits.

We leave this proof as an instructive exercise, in particular observe how the commutation conditions we found in the previous discussion arise as a consequence of the abstract definitions. Conclude that the isomorphism h: TA→A, that we have built in the previous discussion, is the initial T_z-algebra.

From Contravariant Functors to Covariant Functors
Consider the functor ⇒: $\mathbf{C}^{op}\times\mathbf{C}\to\mathbf{C}$, defined in every ccc, that given objects A, B returns the exponent exp(A, B) (with the standard extension to morphisms). We would like to find solutions to equations such as: X = X⇒D, or X = X⇒X. The problem here is that there is no way we can look at λX. X⇒D, or λX. X⇒X as (covariant) functors. In the spirit of the previous Proposition *(Relative Initial Solution)* we will introduce new structures that will allow us to reduce the problem to the *(Initial Solution)* Proposition. In the first place we present the notion of *embedding-projection pair* in an *O-category*.

Definition *(O-category)*
A category **C** is an *O-category* if
(1) every hom-set is an ω-directed complete partial order (i.e. every ω-chain has a lub).
(2) composition of morphisms is a continuous operation with respect to the orders of the hom-sets.

The first relevant application of the notion of embedding-projection pair in domain theory appears in Scott[72]. Wand introduced the notion of O-category.

Example: Cpo is an O-category, ordering the arrows pointwise.

Definition *(Retraction, embedding, projection)*
Let **C** be an O-category, A, B∈**C**.
(1) A *retraction* from A to B is a pair (i,j) such that i: A→B, j: B→A, $j\circ i = id_A$ (we write then A◁B).
(2) An *embedding-projection* from A to B is a pair (i,j) such that i: A→B, j: B→A, $j\circ i = id_A$, $i\circ j \le id_B$ (we write then A⊑B, of course A⊑B implies A◁B).
(3) A *projection* on A is a map p: A→A such that $p\circ p = p$ and $p \le id_A$.

Exercise iDETj: Show that in an embedding-projection pair (i,j), i determines j and j determines i. Show that if (i,j) is a projection pair, then $i\circ j$ is a projection.

Definition *(Category of Embedding-Projection Pairs)*

Let **C** be an O-category. The category $\mathbf{C}^{ep}$ has the same objects as **C** and embedding-projection pairs as morphisms:

$$\mathbf{C}^{ep}(A,B) \triangleq \{(i, j) \mid i: A \to B, j: B \to A, j \circ i = id_A, i \circ j \leq id_B\} .$$

Composition is given by $(i, j) \circ (i', j') = (i \circ i', j' \circ j)$, identities by (id,id).

Proposition

Let **C** be an O-category. Then

(1) $\mathbf{C}^{ep}$ is a category in which all morphisms are monos.

(2) If **C** has a terminal object, if each hom-set **C**(A, B) has a least element $\perp_{A,B}$, and if composition is left-strict (i.e. $f: A \to A' \Rightarrow \perp_{A',A''} \circ f = \perp_{A,A''}$), then $\mathbf{C}^{ep}$ has an initial object.

Proof

(1) Suppose: $(i, j) \circ (i', j') = (i, j) \circ (i'', j'')$. That is $(i \circ i', j' \circ j) = (i \circ i'', j'' \circ j)$. Since i is a mono: $i \circ i' = i \circ i'' \Rightarrow i' = i''$. Therefore, by exercise iDETj, $j' = j''$.

(2) Let 1 be the terminal object in **C**. We show that 1 is initial in $\mathbf{C}^{ep}$. Given $A \in \mathbf{C}$, we first show $(\perp_{1,A}, \perp_{A,1}) \in \mathbf{C}^{ep}(1,A)$. On one hand, $\perp_{A,1} \circ \perp_{1,A} = id_1$ since 1 is terminal, on the other hand $\perp_{1,A} \circ \perp_{A,1} = \perp_{A,A} \leq id_A$ since composition is left strict. By exercise iDETj, there are no other arrows in $\mathbf{C}^{ep}(1,A)$ since $\perp_{A,1}$ is the unique element of **C**(A,1). □

We are now in a position to suggest what the category of embedding-projection pairs is good for. Given a functor $F: \mathbf{C}^{op} \times \mathbf{C} \to \mathbf{C}$ we build a functor $F^{ep}: \mathbf{C}^{ep} \times \mathbf{C}^{ep} \to \mathbf{C}^{ep}$ which coincides with F on objects. In particular the exponent functor is transformed into a functor which is covariant in both arguments. Hence if $F^{ep}D \cong D$ in $\mathbf{C}^{ep}$ then $FD \cong D$ in **C**. In other words we build a related structure, $\mathbf{C}^{ep}$, and a related problem, $F^{ep}D \cong D$, whose solutions can be used for the initial problem. The advantage of the related problem is that we only have to deal with covariant functors and therefore we are in a favorable position to apply the proposition *Initial Solution*. Towards this goal it is natural to look for conditions on **C** that guarantee that $\mathbf{C}^{ep}$ has ω-colimits (we already know that under certain conditions it has an initial object) as well as for conditions on F that guarantee that F^{ep} is ω-cocontinuous. This is the sense of the two following results.

Let **C** be an O-category and $F: \mathbf{C}^{op} \times \mathbf{C} \to \mathbf{C}$ be a functor (the generalization to n arguments is immediate). We say that F is *locally monotone* (*continuous*) if it is monotone (ω-continuous) w.r.t the orders on the hom-sets. There is a standard technique to transform a covariant-contravariant monotone functor on **C** into a covariant functor on $\mathbf{C}^{ep}$. Given $F: \mathbf{C}^{op} \times \mathbf{C} \to \mathbf{C}$ define $F^{ep}: \mathbf{C}^{ep} \times \mathbf{C}^{ep} \to \mathbf{C}^{ep}$ as follows

$$F^{ep}(c, c') \triangleq F(c, c')$$
$$F^{ep}((i, j), (i', j')) \triangleq (F(j, i'), F(i, j'))$$

Exercise: Verify that F^{ep} as defined above is a functor.

The following result points out that the ω-colimit of $\{D_n, (i_n, j_n)\}_{n\in\omega}$ in $\mathbf{C}^{ep}$ can be derived from the ω^{op}-limit of $\{D_n, j_n\}_{n\in\omega}$ in $\mathbf{C}$. One often refers to this situation as limit-colimit coincidence.

Theorem *(Limit-Colimit Coincidence)*
Let $\mathbf{C}$ be an O-category. If $\mathbf{C}$ has ω^{op}-limits then $\mathbf{C}^{ep}$ has ω-colimits.

Proof

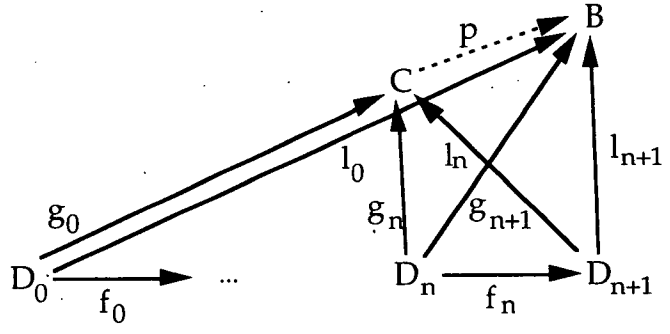
Consider an ω-chain in $\mathbf{C}^{ep}$ $\{D_n, f_n\}_{n\in\omega}$ where we denote with $f_n^+: D_n \to D_{n+1}$ the mono and with $f_n^-: D_{n+1} \to D_n$ the epi. Let $\{C, g_n^-\}_{n\in\omega} = \lim_{\mathbf{C}} \{D_n, f_n^-\}_{n\in\omega}$. We show that D_m can be made into a *cone* for $\{D_n, f_n^-\}_{n\in\omega}$, for all m. Let $h_{mn}: D_m \to D_n$ be the natural way to go from D_m to D_n, it is defined as follows:

$h_{mn} \triangleq id_{D_m}$, if m=n. $\quad h_{mn} \triangleq f_n^- \circ \ldots \circ f_{m-1}^-$, if m>n. $\quad h_{mn} \triangleq f_{n-1}^+ \circ \ldots \circ f_m^+$, if m<n.

Verify that $\{D_m, h_{mn}\}_{n\in\omega}$ is a cone for $\{D_n, f_n^-\}_{n\in\omega}$. Hence a unique map $g_m^+: D_m \to C$ is determined, such that $(g_m^+, g_m^-): D_m \to C$ in $\mathbf{C}^{ep}$ (this justifies our notation). We leave to the reader to check that $g_m^- \circ g_m^+ = id_{D_m}$. Let us verify $g_m^+ \circ g_m^- \leq id_{D_{m+1}}$. Observe: $g_m^+ \circ g_m^- = g_{m+1}^+ \circ f_m^+ \circ f_m^- \circ g_{m+1}^- \leq g_{m+1}^+ \circ g_{m+1}^-$. Hence $\{g_m^+ \circ g_m^-\}_{m\in\omega}$ is a chain and we write $k = \bigsqcup_{m\in\omega} g_m^+ \circ g_m^-$. In order to prove that $k=id_C$ it is enough to remark that k is a cone endomorphism over $\{C, g_m^-\}_{m\in\omega}$ as:

$g_m^- \circ k = g_m^- \circ \bigsqcup_{i\in\omega} g_i^+ \circ g_i^- = g_m^- \circ \bigsqcup_{i\geq m} g_i^+ \circ g_i^- = \bigsqcup_{i\geq m} g_m^- \circ g_i^+ \circ g_i^- = \bigsqcup_{i\geq m} h_{im} \circ g_i^- = g_m^-$.

It remains to show that $\{C, g_m\}_{m\in\omega}$ is a colimit. Let $\{B, l_m\}_{m\in\omega}$ be another cocone. Define: $p^+ \triangleq \bigsqcup_{m\in\omega} l_m^+ \circ g_m^-: C \to B$, and $p^- \triangleq \bigsqcup_{m\in\omega} g_m^+ \circ l_m^-: B \to C$. Verify $p: C \to B$ in $\mathbf{C}^{ep}$.
Moreover p is a morphism of cocones between $\{C, g_m\}_{m\in\omega}$ and $\{B, l_m\}_{m\in\omega}$. Finally suppose q is another such morphism; then referring to the k defined above:
$(q^+, q^-) = (q^+ \circ k, k \circ q^-) = (q^+ \circ (\bigsqcup_{m\in\omega} g_m^+ \circ g_m^-), (\bigsqcup_{m\in\omega} g_m^+ \circ g_m^-) \circ q^-) =$
$(\bigsqcup_{m\in\omega} q^+ \circ g_m^+ \circ g_m^-, \bigsqcup_{m\in\omega} g_m^+ \circ g_m^- \circ q^-) = (\bigsqcup_{m\in\omega} l_m^+ \circ g_m^-, \bigsqcup_{m\in\omega} g_m^+ \circ l_m) = (p^+, p^-)$. Here is a diagram of the situation:



□

Exercise: With reference to the previous proof show that the cocone $\{C, g_m\}_{m\in\omega}$ is a colimit in in $\mathbf{C}^{ep}$ iff $\bigsqcup_{m\in\omega} g_m^+ \circ g_m^- = id_C$.

Note *(which ω^{op}-limits are needed ?)*

It is often the case that the category of domains under consideration does not have ω^{op}-limits but only limits of ω^{op}-diagrams whose arrows are the projection component of an embedding-projection pair. Note that this is enough to apply the technique for the solution of domain equations described here.

The following result relates local continuity and preservation of ω-colimits.

Proposition *(Induced Functor)*

Let **C** be an O-category with ω^{op}-limits. If $F: \mathbf{C}^{op}\times\mathbf{C}\to\mathbf{C}$ is a locally continuous functor then $F^{ep}: \mathbf{C}^{ep}\times\mathbf{C}^{ep}\to\mathbf{C}^{ep}$ preserves ω-colimits.

Proof

We have already observed that if F is locally monotone then F^{ep} is a functor. Let $\{(D_n, E_n), (f_n, g_n)\}_{n\in\omega}$ be an ω-diagram in $\mathbf{C}^{ep}$ with colimit $\{(D, E), (h_n, k_n)\}_{n\in\omega}$ built as in the previous theorem.

To show that the cone $\{F^{ep}(D, E), F^{ep}(h_n, k_n)\}_{n\in\omega}$ is a colimit for $\{F^{ep}(D_n, E_n), F^{ep}(f_n, g_n)\}_{n\in\omega}$ it is enough to verify that:

$\bigsqcup_{n\in\omega} F(h_n^-, k_n^+)\circ F(h_n^+, k_n^-) = id_{F(D, E)}$, this is proven as follows

$\bigsqcup_{n\in\omega} F(h_n^-, k_n^+)\circ F(h_n^+, k_n^-) = \bigsqcup_{n\in\omega} F(h_n^+\circ h_n^-, k_n^+\circ k_n^-) =$

$F(\bigsqcup_{n\in\omega} h_n^+\circ h_n^-, \bigsqcup_{n\in\omega} k_n^+\circ k_n^-) = F(id_D, id_E) = (id_{F(D)}, id_{F(E)}) = id_{F(D, E)}$. □

To summarize the method, given: (1) an O-category **C** such that the hom-sets have a least element, composition is left strict, and **C** has (certain) ω^{op}-limits, (2) A locally continuous functor $F: \mathbf{C}^{op}\times\mathbf{C}\to\mathbf{C}$, *we can apply the previous constructions and build:* (3) the category $\mathbf{C}^{ep}$ which has initial object and ω-colimits, (4) the functor $F^{ep}: \mathbf{C}^{ep}\times\mathbf{C}^{ep}\to\mathbf{C}^{ep}$ which preserves ω-colimits, *therefore:* (5) we find an initial solution for $F^{ep}(X, X)\cong X$ in $\mathbf{C}^{ep}$, *which also gives:* (6) a solution for the equation $F(X, X)\cong X$ in **C**.

We will show in the next section of this chapter how to apply this method to the ccc of bifinite domains. For the moment we can practice on the category of cpos and continuous maps.

Exercise ω^{op}-CPO: Show that the category **Cpo** has a terminal object and ω^{op}-limits.

Exercise Exp-Prod-Cont: Verify that the product and exponent functors

Prod: $\mathbf{Cpo}\times\mathbf{Cpo}\to\mathbf{Cpo}$ Exp: $\mathbf{Cpo}^{op}\times\mathbf{Cpo}\to\mathbf{Cpo}$

are locally continuous.

Exercise $\lambda\beta\eta$+SP: Show the existence of a non-trivial domain D such that: $D\cong D\times D\cong D\to D$. Hint: consider the system $D\cong D\to E$, $E\cong E\times E$.

Exercise λ_V-λ_L : Let $(\ )_\perp$ be the lifting functor (see chapter 4). Show that the equations $D \cong D \to (D)_\perp$ and $D \cong (D)_\perp \to (D)_\perp$ have a non trivial initial solution in $\mathbf{Cpo}^{ep}$

Exercise D_∞: Explain how to build two non-isomorphic, non-trivial solutions of the equation $D \cong D \to D$.

2. Universal Domains

We discuss a technique for the construction of a "universal" domain and we apply it to the category of bifinites and continuous maps. In this section by bifinite domain we intend the ω-bifinite (or SFP) described in chpt. 3.

Universal Homogeneous Domains in ω-Algebroidal Categories

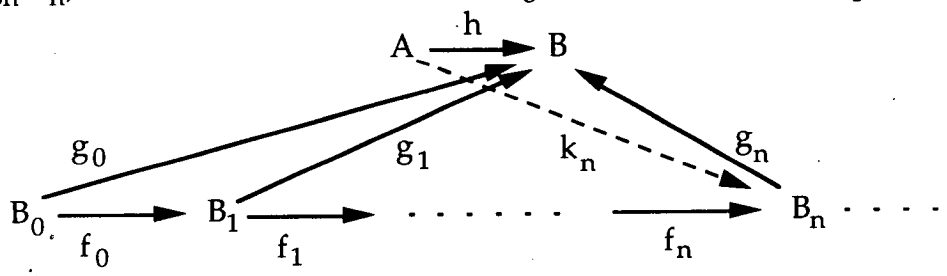
In the first place we introduce the notion of algebroidal category. This generalizes to categories the notion of algebraicity already considered for domains.

Definition *(Category of monos)*

A category **K** is a *category of monos* if every arrow of **K** is a monomorphism.

Definition *(Compact Object)*

Let **K** be a category of monos. An object $A \in \mathbf{K}$ is *compact* if for each ω-chain $\{B_n, f_n\}_{n\in\omega}$ with colimit $\{B, g_n\}_{n\in\omega}$ and any $h: A \to B$ there exists n and $k_n: A \to B_n$ such that $h = g_n \circ k_n$, for some n. We denote with K_o the collection of compact objects.



Notice that k_n is unique, as **K** is a category of monos, and that for any p larger than n, $k_p \triangleq f_{n,p} \circ k_n$ satisfies the specification in the definition, that is $h = g_p \circ k_p$.

Example: Sets with injections form a category of monos.

Definition *((ω-)Algebroidal Category)*

A category of monos **K** is *algebroidal* if

(1) K has an initial object.

(2) Every object is the colimit of an ω-chain of compact objects.

(3) Every ω-chain of compact objects has a colimit.

An algebroidal category is *ω-algebroidal* if the collection of compact objects, K_o, is countable up to isomorphism and so is the hom-set between any two compact objects.[9]

Exercise NOT-ALG: Let **S** be the category of Scott domains, i.e. bounded complete algebraic cpos and continuous maps. Show that $\mathbf{S}^{ep}$ is not an algebroidal category. How would you modify the definition in order to include $\mathbf{S}^{ep}$ among the algebroidal categories ? **Hint:** A directed set is a preorder (I, ≤) such that I is a directed set. A directed diagram in a category **C** is a functor D: I→**C**. Show that: (a) $\mathbf{S}^{ep}$ has colimits of directed diagrams; (b) each object is the colimit of a directed diagram of compact objects.

Next we define the notion of universal object. In particular we will be interested in universal, *homogeneous* objects, as they are determined up to isomorphism. In this section we follow quite closely Droste&Göbel[90]. More generally terminology and techniques used in this section are clearly indebted to model theory.

Definition

Let U be an object in a category **K** of monos, and **K*** be a full subcategory of **K**. Then:

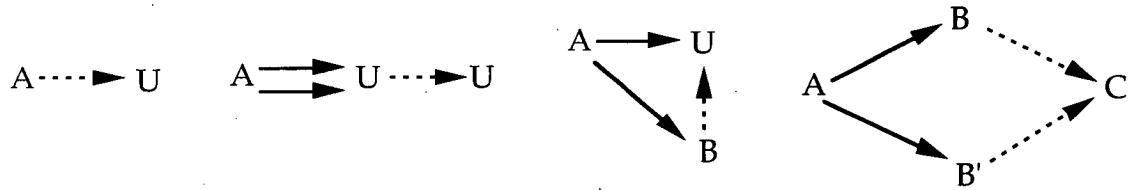
(1) U is *K*-universal* if $\forall A \in K^*.\exists f: A \to U$

(2) U is *K*-homogeneous* if $\forall A \in K^*.\ \forall f, g: A \to U.\ \exists h: U \to U.\ (h \circ g = f)$

(3) U is *K*-saturated* if $\forall A, B \in K^*.\ \forall f: A \to U.\ \forall g: A \to B.\ \exists h: B \to U.\ (h \circ g = f)$

(4) K* has the *amalgamation property* if

$\forall A, B, B' \in K^*.\forall f: A \to B.\forall f': A \to B'.\exists C \in K^*.\exists g: B \to C.\exists g': B' \to C.\ (g \circ f = g' \circ f')$.[10]



Universality Homogeneity Saturation Amalgamation

[9] Terminology: following Banaschewski and Herrlich we call ω-algebroidal what is called algebroidal by Smyth and Plotkin. In all cases considered in these notes the collection of compact objects is countable up to isomorphism. However thinking of ordinals as a category one sees that there are interesting cases where the collection of compact objects is not countable.

[10] Remark that in all these definitions we require the existence of certain morphisms but *not* their universality.

Theorem *(Droste&Göbel[90])*

Let K be an ω-algebroidal category of monos. The following are equivalent:

(1) There is a K-universal, K_0-homogeneous object.

(2) There is a K_0-saturated object.

(3) K_0 has the amalgamation property.

Moreover a K-universal, K_0-homogeneous object is uniquely determined up to isomorphism.

The proof of this theorem is an immediate consequence of the following lemmas. The main difficulty lies in the construction of a K_0-saturated object while relying on the hypothesis of K_0-amalgamation (see K_0-AMLG $\Rightarrow$ K_0-SAT).

Lemma *(K_0-SAT-ISO)*

Let **K** be an algebroidal category of monos and let U, V be K_0-saturated. Then:

$\forall A \in K_0 . \forall f: A \to U . \forall g: A \to V . \exists i: U \to V$ iso. $(i \circ g = f)$.

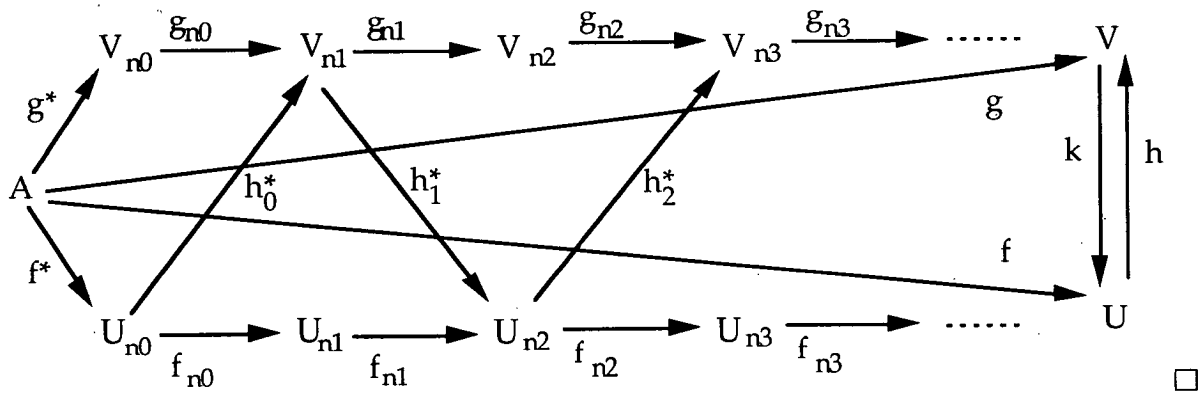
Proof

Refer to the diagram below. $\{(U_i, f_i)\}_{i \in \omega}$ and $\{(V_j, g_j)\}_{j \in \omega}$ are ω-diagrams of compact objects whose colimits are U and V respectively.

• Given f: A→ U and g: A→V consider f*: A→ U_{n0} and g*: A→V_{n0} that exist by the compactness of A.

• Next use K_0-saturation and the compactness definition to go from U_{no} to V_{n1} via h^*_0. Proceed inductively in this way determining U_{n2}, V_{n3}, etcetera.

It is then possible, using the h^*_i, to see V as (the object of) a cocone for $\{(U_i, f_i)\}_{i \in \omega}$ and U as (the object of) a cocone for $\{(V_j, g_j)\}_{j \in \omega}$, by which the existence of the isomorphisms h and k which commute with f and g follows.



□

Lemma*(EQUIV)*

Let **K** be an algebroidal category of monos.

(1) For any object U the following are equivalent:

(a) U is K-universal and K_0-homogeneous.

(b) U is K_0-universal and K_0-homogeneous.

(c) U is K_0-saturated.

(2) A K-universal, K_o-homogeneous object is determined up to isomorphism.
(3) If there is a K-universal and K_o-homogeneous object then K_o has the amalgamation property.

Proof

(1) (a)⇒(b): immediate, by definition.

(b)⇒(c): Let $A, B \in K_o$, $f: A \to U$, $g: A \to B$. By K_o-universality $\exists g': B \to U$. By K_o-homogeneity $\exists h: U \to U$. $(h \circ g' \circ g = f)$. So $h \circ g'$ gives saturation.

(c)⇒(a): Since there is an initial object O, U is K_o-universal via saturation applied to the (unique) maps: $f: O \to U$ and $g: O \to A$. U is also K_o-homogeneous by an instance of lemma (K_o-SAT-ISO). It remains to show that U is K-universal. Let $A \in K$ and let $\{(A_i, f_i)\}_{i \in \omega}$ be an ω-chain in K_o whose colimit is A. Take advantage of K_o-saturation to build a cocone with object U for this ω-chain. Then there is a map from A to U.

(2) Apply lemma (K_o-SAT-ISO) with A=O.

(3) Let $A, B, B' \in K_o$, $f: A \to B$, $f': A \to B'$. By K_o-universality $\exists h: B \to U$. By K_o-saturation $\exists h': B' \to U. (h \circ f = h' \circ f')$. Now consider an ω-chain in K_o whose colimit is U and use the compactness of B and B' to factorize h and h' along some element of the ω-chain. □

We are now going to use (for the first time) the countability conditions that distinguish an ω-algebroidal category from an algebroidal one.

Lemma *(K_o-AMLG ⇒ K_o-SAT)*

Let **K** be an ω-algebroidal category of monos. If K_o has the amalgamation property then it is possible to build a K_o-saturated object.

Proof

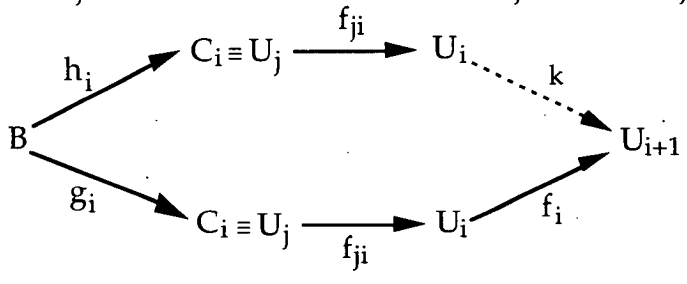
Use the hypothesis that K is ω-algebroidal to build an enumeration up to isomorphism of the compact objects $H_o \triangleq \{A_i \mid i \in \omega\}$ and an enumeration of all quintuples $M_o \triangleq \{(B_i, C_i, g_i, h_i, j_i) \mid i \in \omega\}$ where: $B_i, C_i \in H_o$, $g_i, h_i: B_i \to C_i$ and $j_i \in \omega$, such that each quintuple occurs infinitely often. In this way, given $j \in \omega$, we can always find $i \in \omega$ such that $j_i = j \leq i$.

Next build an ω-chain $\{(U_i, f_i)\}_{i \in \omega}$ such that $U_i \in H_o$ and with the properties:

(1) $\forall i \in \omega. \exists k_i: A_i \to U_i$.

(2) Given i consider the corresponding quintuple in the enumeration:

if $j \equiv j_i \leq i$ and $U_j \equiv C_i$ then $\exists k: U_i \to U_{i+1}. (k \circ f_{ji} \circ h_i = f_i \circ f_{ji} \circ g_i)$.



Two important facts are hidden under these conditions: (a) when we build the

ω-chain $\{(U_i, f_i)\}_{i\in\omega}$, we will have to satisfy at most two constraints as specified by (1) and (2). (b) If we have f, g: B→C with B, C∈H_o, and C≡U_j then (B,C, g, h, j) will appear infinitely often in the enumeration so we can find an i such that: (B, C, g, h, j) ≡ (B_i, C_i, g_i, h_i, j_i) and ($j \equiv$)$j_i \le i$.

The idea is to take U as the colimit of the ω-chain $\{(U_n, f_n)\}_{n\in\omega}$. While condition (1) is natural, condition (2) may seem rather obscure. First observe that if we just want to build a K_o-universal object, that is satisfy condition (1), then it is enough to set $U_0 \triangleq A_0$ and proceed inductively using the amalgamation property on the (uniquely determined) maps f: O→U_n and g: O→A_{n+1}. So, given (EQUIV), condition (2) has to do with the fact that we want U to be K_o-saturated.

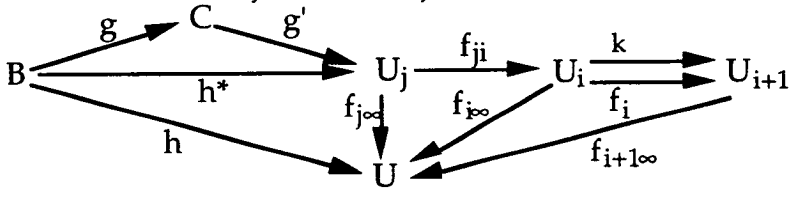
We now show how to use condition (2) to get K_o-saturation. Let B, C∈H_o and g: B→C, h: B→U. By (I) and B∈K_o we have (see diagram below):

$\exists j.(g'\colon C\to U_j, h^*\colon B\to U_j, h = f_{j\infty}\circ h^*)$.

Let $g^* = g\circ g'$. Choose i large enough so that:

$j\le i \wedge (B, U_j, g^*, h^*, j) \equiv (B_i, C_i, g_i, h_i, j_i)$.

By (II) there is k: $U_i\to U_{i+1}$.($k \circ f_{ji} \circ h^* = f_i \circ f_{ji} \circ g^*$). From this, saturation follows.



Finally we show how to build the ω-chain $\{(U_i, f_i)\}_{i\in\omega}$. Set $U_0 = A_0$, the first element in the enumeration H_o. Next suppose to have built U_i and consider A_{i+1}. As observed above there are f: O→U_i and g: O→A_{i+1}. By amalgamation we get, for some U_i', two maps f': U_i→U_i' and g': A_{i+1}→U_i'.

- If we do not satisfy: $j_i \le i$ and $U_{j_i} \equiv C_i$ then it is enough to choose an object U_{i+1} in H_o isomorphic to U_i'. The map from A_{i+1} to U_{i+1} is then immediately obtained by composition.
- On the other hand if $j \equiv j_i \le i$ and $U_j \equiv C_i$ then apply amalgamation to: $h_i \circ f_{ji} \circ f'$ and $g_i \circ f_{ji} \circ g'$ obtaining k: U_i'→U_{i+1}' and k': U_i'→U_{i+1}'. It just remains to select U_{i+1} isomorphic to U_{i+1}' and in H_o. □

Application to Bifinite Domains

In this section we verify that **Bif^{ep}** is an ω-algebroidal category with the amalgamation property. Therefore it has a universal homogeneous object. This result depends on the construction of ω-colimits in **Bif^{ep}** (and ω^{op}-limit in **Bif**). This important construction is described following the ideas presented in the first section of this chapter.

Theorem *(**Bif**^{ep} has a Universal Homogeneous Object)*

Let **Bif**^{ep} be the category of bifinite domains and embedding projection pairs.

(1) **Bif**^{ep} is an ω-algebroidal category and the collection of compact objects has the amalgamation property.

(2) **Bif**^{ep} has a universal homogeneous object.

Proof

(1) We decompose the statement in more elementary steps.

(a) **Bif**^{ep} is a category of monos with initial object.

We have seen that under the following hypothesis **C**^{ep} is a category of monos with initial object: **C** has a terminal object, the hom-sets have a least element and composition is left strict. Verify that these conditions are satisfied by **Bif** !

(b) Let D∈ Bif. Then D is compact in **Bif**^{ep} iff the cardinality of D is finite.

(⇒) Let $\{q_n\}_{n\in\omega}$ be the chain of projections of finite image such that $\sqcup\{q_n\}_{n\in\omega} = id_D$. Consider the identity morphism on D, from the definition of compact object we conclude $\exists n.\ id \le q_n$, so im(id) = D is finite.

(⇐) After some manipulations the problem is reduced to the following: let $\{q_n\}_{n\in\omega}$ be a chain of projections on an object E such that $\sqcup\{q_n\}_{n\in\omega} = id_E$ and p another projection on D such that im(p) is finite. Now observe that if im(p) is finite and p(x) = x then x is compact in D. From this it immediately follows that p is compact in D→D. Therefore $p \le \sqcup\{q_n\}_{n\in\omega} = id_E \Rightarrow \exists n.\ p \le q_n$.

(c) Each object in **Bif**^{ep} is an ω-colimit of compact objects.

This is a consequence of (2) and the definition of bifinite object.

(d) Each ω-diagram of (compact) objects in **Bif**^{ep} has a colimit.

Consider an ω-chain in **Bif**^{ep} $\{D_n, f_n\}_{n\in\omega}$ where we denote with $f_n^+: D_n \to D_{n+1}$ the injection and with $f_n^-: D_{n+1} \to D_n$ the surjection. Define:

$$D \triangleq \{\alpha: \omega \to \cup_{n\in\omega} D_n \mid \alpha(n)\in D_n \wedge f_n^-(\alpha(n+1)) = \alpha(n)\}.$$

with the pointwise ordering: $\alpha \le_D \beta$ iff $\forall n\in\omega.\ \alpha(n) \le_{D_n} \beta(n)$. Verify that this makes D into a cpo. First we show that D can be made into a *cocone*. Let $h_{ij}: D_i \to D_j$ be the natural way to go from D_i to D_j, it is defined as follows:

$h_{ij} \triangleq id_{D_i}$, if i=j. $\quad h_{ij} \triangleq f_j^- \circ \ldots \circ f_{i-1}^-$, if i>j. $\quad h_{ij} \triangleq f_{j-1}^+ \circ \ldots \circ f_i^+$, if i<j.

Next define $g_n: D_n \to D$ as:

$g_n^-(d) \triangleq \alpha(n) \qquad g_n^+(d) \triangleq \lambda i\in\omega.\ h_{ni}(d)$

Verify that (g_n^-, g_n^+) is a morphism and prove $\sqcup_{n\in\omega} g_n^+ \circ g_n^- = id$. Use this fact to infer that D is a bifinite.

The verification that $\{D, g_n\}_{n\in\omega}$ is a colimit is just an instance of the general technique that we described for O-categories. Roughly let $\{E, h_n\}_{n\in\omega}$ be another cocone and consider the map k: D→E, $k \triangleq (\sqcup_{n\in\omega} h_n^+ \circ g_n^-, \sqcup_{n\in\omega} g_n^+ \circ h_n^-)$.

(e) **Bif**^{ep} has the amalgamation property.

Let us consider three finite posets (E, ≤), (D_1, $\le_1$), (D_2, $\le_2$) with maps $h_i: E \to D_i$, i∈{1,2}, in **Bif**^{ep}. W.l.o.g. assume $E = D_1 \cap D_2$, then:

$\forall e, e'\in E.\ (e \le e' \Leftrightarrow e \le_1 e' \Leftrightarrow e \le_2 e')$.

Now we define the amalgam as the set $F \triangleq E \cup (D_1 \setminus E) \cup (D_2 \setminus E)$ where:

$f \leq_F f' \Leftrightarrow$ (1) $f, f' \in D_i$, $f \leq_i f'$ for $i \in \{1, 2\}$ or (2) $\exists e \in E.(f \leq_i e \leq_j f')$ for $i \neq j$, $i, j \in \{1, 2\}$.

Verify that $\leq_F$ is a partial order. We are left with the definition of the morphisms $k_i: D_i \to F$, $i \in \{1, 2\}$. Take the inclusions for k_i^+. Define:

$k_1^-(f) \triangleq$ if $f \in D_1$ then f else $h_2^-(f)$. k_2^- is defined symmetrically. Verify: (a) k_i is a morphism in Bif^{ep}. (b) $k_1 \circ h_1 = k_2 \circ h_2$.

(2) Bif^{ep} is an ω-algebroidal category, therefore we can apply theorem 1.1. □

Note: Consider the proof of (d). Observe that: (1) We never used the fact that D_n is a compact object. (2) A subset of the arguments given there proves that Cpo has ω^{op}-limits. (3) As a matter of fact we show:

$$\mathrm{colim}_{\mathrm{Cpo}^{ep}} \{(D_n, f_n)\}_{n \in \omega} = \lim_{\mathrm{Cpo}} \{(D_n, f_n^-)\}_{n \in \omega}$$

that is an instance of the general theorem on the limit-colimit coincidence.

3. Operator Representation

In this section we are interested in the problem of representing "subdomains" of a domain D as certain functions over D. In particular we concentrate on *retractions* and *projections*, the idea being that subdomains are represented by the image of these maps. In the continuous case not every retraction (or projection) corresponds to a domain (i.e. an ω-algebraic cpo). For this reason one focuses on the collection of *finitary* retractions, which are by definition those retractions whose image forms a domain.

The theory is simpler when dealing with (finitary) projections. Then it is not difficult to show that the collection of finitary projections FP(D) over a bifinite D is again a bifinite. In other words the collection of "subdomains" of a bifinite domain can be given again a bifinite domain structure. This powerful fact is partially exploited when we discuss representations and solutions of domain equations over FP(U), for U universal domain.

Retractions

The collection of retractions on a cpo D, Ret(D), is the collection of fixpoints of the functional $\lambda f.f \circ f$ and the image of a retraction on D, r(D), coincides with the collection of its fixpoints. Hence the general results on fixpoints can be immediately applied. We will see however that under suitable hypotheses something more can be said about the structure of Ret(D) and r(D).

Fact

Let D be a cpo. Then:

(1) If $f: D \to D$ is a continuous map then $\mathrm{Fix}(f) \triangleq \{ d \in D \mid f(d) = d \}$ is a cpo.

(2) $\mathrm{Ret}(D) = \mathrm{Fix}(\lambda f{:}D \to D.\ f \circ f)$ is a cpo.

(3) If $r \in \mathrm{Ret}(D)$ then $r(D) = \mathrm{Fix}(r)$ is a cpo.

Proposition *(compacts in r(D))*

Let D be an algebraic cpo and $r \in Ret(D)$. Then $r(D)_0$, the collection of compacts in r(D), can be characterized as follows: $r(D)_0 = \{rd \mid d \in D_0 \wedge d \leq rd\}$.

Proof

Suppose $d \in D_0$ and $d \leq rd$. Let $X \subseteq r(D)$ directed. Then :

$$d \leq rd \leq \bigsqcup X \Rightarrow \exists y \in X.(d \leq y) \Rightarrow \exists y \in X.(rd \leq y = ry).$$

Vice versa suppose $ry \in r(D)_0$. Since D is algebraic $ry = \bigsqcup\{x \in D_0 \mid x \leq ry\}$. So:

$$r(ry) = ry = r(\bigsqcup\{x \in D_0 \mid x \leq ry\}) = \bigsqcup\{rx \mid x \in D_0 \wedge x \leq ry\}.$$

Since $ry \in r(D)_0$, we have $\exists z.\ ry = rz \wedge z \in D_0 \wedge z \leq rz$. This z gives the desired representation of ry. □

Notes

(1) If D is a bounded complete cpo and $r \in Ret(D)$ then r(D) is bounded complete. Let $X \subseteq r(D)$ and suppose $y \in r(D)$ is a bound for X. Then X is bounded in D and therefore $\exists \bigsqcup_D X$. We show: $\exists \bigsqcup_{r(D)} X = r(\bigsqcup_D X)$.

- $\forall x \in X.(x \leq \bigsqcup_D X) \Rightarrow \forall x \in X.(rx = x \leq r(\bigsqcup_D X))$. So $r(\bigsqcup_D X)$ is an upper bound.
- If y is an upper bound for X in r(D) then it is also an upper bound for X in D so $\bigsqcup_D X \leq y$. This implies $r(\bigsqcup_D X) \leq ry = y$. So $r(\bigsqcup_D X)$ is the lub in r(D).

(2) If p is a projection (i.e. $p \circ p = p$, $p \leq id$) then $p(D)_0 = p(D) \cap D_0$.

(3) If c is a closure (i.e. $c \circ c = c$, $c \geq id$) then $c(D)_0 = c(D_0)$.

Excursus *(finitary retractions)*

Let D be an (ω-)algebraic cpo and $r \in Ret(D)$. Can we conclude that r(D) is again an (ω-)algebraic cpo? The answer is NO. In general it can only be shown that r(D) is a *continuous* cpo (see chapter 3). The following is an example of an ω-algebraic complete total order with a projection on it whose image is not algebraic:

$$D \triangleq (\{[0, q] \mid q \in \mathbb{Q}\} \cup \{[0, r[\mid r \in \mathbb{R} \cup \{+\infty\}\}, \subseteq)$$
$$p([0, q]) \triangleq [0, q[,\ p([0, r[) \triangleq [0, r[\ .$$

When considering the collection Ret(D) things get even worse. For example it has been shown by Ershov (see exercise 18.4.10 in Barendregt[84]) that the collection of retractions over Pω is not a continuous lattice, hence a fortiori not the image of a retraction (this also shows that the collection of fixpoints of a continuous function does not need to be a continuous cpo, as $Ret(D) = Fix(\lambda f.\ f \circ f)$.

We will consider retractions again in the context of stable domain theory. For the time being we will concentrate on the simpler case of finitary projections.

Finitary Projections over Bifinite Domains

Let D be a bifinite domain. The notion of finitary projection over D provides an adequate representation of the idea of subdomain, moreover the collection of finitary projections over D, FP(D), is again a bifinite domain. This is a powerful result that has application for instance to the interpretation of higher-order calculi

(see chapter 6). The following notion of normal subposet is useful in studying projections.

Definition *(Normal Subposet)*

Let $(P, \leq)$ be a poset. $N \subseteq P$ is a *normal subposet* if $\forall x \in P.\ \downarrow x \cap N$ is directed. We denote with $N(P)$ the collection of normal subposets of P ordered by inclusion.

Proposition *(Finitary Projections and Bifinites)*

Let $D \in$ Bif. Then:

(1) There is an isomorphism between the collection of normal subposets of the compact elements and the finitary projections over D: $N(D_o) \cong FP(D)$.

(2) FP(D) is an ω-algebraic complete lattice.

Proof

Preliminary Remarks: (A) We know that $p(D)_o = p(D) \cap D_o$. (B) If p is a projection then $p(D) \in N(D)$ as $\downarrow x \cap p(D) = \downarrow p(x) \cap p(D)$. (C) If p is a finitary projection then $p(D)_o \in N(D_o)$. Use the hypothesis p(D) algebraic to show: $\forall x \in D.\ \downarrow x \cap p(D)_o = \downarrow p(x) \cap p(D)_o$ that is directed.

(1) If p is a finitary projection then define: $N_p \triangleq p(D)_o$. This is a normal subposet of D_o by (A), (C). Vice versa if $N \in N(D_o)$ then define: $p_N \triangleq \lambda d.\ \sqcup(\downarrow d \cap N)$. This is well defined because $\downarrow d \cap N$ is directed. Next verify that these two transformations define an isomorphism between FP(D) and $N(D_o)$.

(2) Left as exercise. □

Exercises PP: (1) Show that if D is an algebraic bounded complete cpo then $FP(D) \subseteq D \to D$. *Hint:* given $f: D \to D$ consider: $X_f \triangleq \{x \in D_o \mid x \leq fx\}$ and define $N_f \triangleq U^*(X_f)$. The set N_f corresponds to a finitary projection p_{N_f}. Set $\pi: (D \to D) \to (D \to D)$ as $\pi(f) \triangleq p_{N_f}$. Verify π is a finitary projection whose image is FP(D).

(2) Find a bifinite for which (1) fails. Hint: it is enough to consider a bifinite D with five elements and observe that there is a map $f: D \to D$ such that $\downarrow f \cap FP(D)$ is not directed.

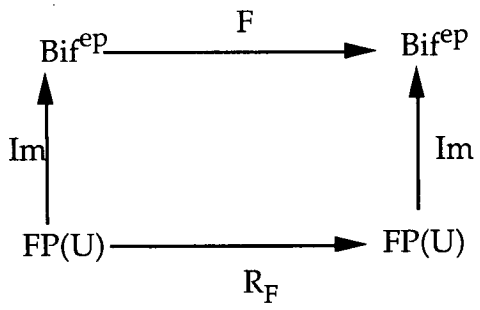
Representing Operators

Let U be a universal domain for some relevant category of domains, say **Bif**. Then each domain can be embedded into U and, therefore, it can be represented by a (finitary) projection over U. Furthermore it can be shown that certain basic operators over domains can be adequately represented as continuous transformations over FP(U). As a fall-out one gets a technique to solve domain equations via the standard Kleene least-fixed point theorem.

Observe that there is an injective function Im from the the poset FP(U) to the category $\mathbf{Bif}^{ep}$: $Im \triangleq \lambda p \in FP(U).\ p(U)$. This is immediately extended to a functor.

Let F be an endofunctor on $\mathbf{Bif}^{ep}$. The *representation problem* consists in finding a continuous transformations R_F on FP(U) such that the following

diagram commutes.[11]



Proposition

Product and Exponent are representable.

Proof

In showing that **Bif** is a ccc, one implicitly uses the fact that if $p \in FP(D)$ and $q \in FP(D)$ then $\lambda(d, e).(p(d), q(e)) \in FP(D \times E)$ and $\lambda f.\lambda d.q(f(p(d))) \in FP(D \to E)$ (actually one can repeat the same argument for retractions and projections). If U is a universal (homogeneous) domain for Bif^{ep} then we may assume the existence of the embedding projection pairs:

$(\lambda(u, u').\ <u, u'>, \lambda u.\ (\pi_1(u), \pi_2(u))): U \times U \to U$ $\quad$ $(i, j): (U \to U) \to U$.

It just remains to combine the two ideas to define the operators representing product and exponential:

$\lambda(p, q).\ \lambda u.\ <p(\pi_1(u)), q(\pi_2(u))>$ $\quad$ $\lambda(p, q).\ \lambda u.\ j(q \circ i(u) \circ p)$. $\quad$ □

Note: It is good to keep in mind that Im is not an equivalence of categories between FP(U) and $\mathbf{Bif}^{ep}$ as FP(U) is just a poset category. This point is actually important when one interprets second order types (chpt. 6).

Exercises REP: (1) Verify in detail that we can apply Tarski fixed point theorem to the domains FP(U) in order to get initial solutions of domain equations in $\mathbf{Bif}^{ep}$.

(2) Consider the representation problem for the operators of coalesced sum and lifting.

(3) Consider the representation problem for (Finitary) Retractions.

[11] It is routine to formulate the representation problem for functors of n arguments.

6. Interpretation of Dependent and Second Order Types

Contents: 1. Domain Theoretic Constructions, 2. A Calculus of Dependent and Second Order Types, 3. Interpretation.

The main goal of this chapter is to show how to interpret 'dependent' and 'second order' types in the framework of traditional domain theory (chapter 8 will mention another approach based on realizability).

Let T be a category whose objects are viewed as types. This category contains atomic types like the singleton type 1, the type Int representing integer computations, and the type Bool representing boolean valued computations. The collection T is also closed w.r.t. certain data type constructions, for example if A and B are types then we can form new types by making a "product type" A×B, a "sum type" A+B, and an "exponent type" A→B .[12]

In first approximation by *dependent type* we mean a family of types indexed over another type A. We represent this family as a transformation F from A into the collection of types T

F: A→T *(intuition of dependent type)*

As an example of dependent type one can think of a family Prod.Bool: Int→T that given an integer n returns the type Bool×...×Bool (n times).

If the family F is indexed over the collection of all types T then we are in the realm of *second order types*

F: T→T *(intuition of second order type)*

As an example of a second order type one can think of a family Fun: T→T that given a type A returns the type A→A of functions over A.

If types, and the collection of types T, can be seen as categories then we can think of dependent and second order types as functors. Let us warn the reader that in this preliminary discussion we are considering a simplified situation. In general we want to combine dependent and second order types and the domain of the family may turn out to be a category different from T. For example one may consider the family Poly.Prod: T×Int→T that takes a type α, an integer n, and returns the type α×...×α (n times).

[12] Brackets are needed as at this point we prefer not to be too precise about the categorical properties of the constructions.

Probably the most familiar example of "dependent type" arises in first order logic. If A(x) is a first order formula depending on the variable x then we can think of A(x) as a family of propositions indexed over the universe of terms U, roughly A:U→Prop. This example suggests the basic motivation behind dependent types, they are a linguistic construct that allows to connect values to truth-value. Hence it is not surprising that dependent types have appeared in several type systems (or generalized logics) such as DeBrujin's Automath, Martin-Löf's Type Theory, and Edinburgh LF.

Keeping in mind the analogy with logic let us consider a formula, A[B], parametric in another formula B. Then we can think of A[B] as a family of propositions indexed over the universe of propositions, roughly A: Prop→Prop. Observe that in this way we can express the idea of parametricity of a formula in another formula. Of course this form of parametricity is already available in any logic but what is often missing is the ability to "quantify" over the parameters. Second order types appear in a rather pure form in Girard system F (or a system of natural deduction for minimal, implicative, propositional second order logic), of course they also appear in second order logic or in the Calculus of Constructions but there they are combined with dependent types and more.

Given a formula A(x) depending on x we can quantify over it and get the formulae $\forall x.A$, and $\exists x.A$. Analogously, in the interpretation, given a family A: U→Prop we can obtain two new proposition $\forall_U A$, and $\exists_U A$ where we understand $\forall_U$ as an inf or product, and $\exists_U$ as a sup or sum.

In general given a family of types F: $\mathbf{C}\to\mathbf{T}$ indexed over a category $\mathbf{C}$ (e.g. think of dependent and second order types) we are interested in building two new types that we may denote, respectively, with $\Pi_{\mathbf{C}}F$ and $\Sigma_{\mathbf{C}}F$, and that correspond respectively to product and sum of the family F.

The main problem that we consider in this section is to provide a concrete *domain theoretic interpretation* of these notions. In particular we build a category of domains that is "closed" under the constructions of (certain) indexed products, and (certain) indexed sums. The first simple idea is to interpret types as domains of a given category $\mathbf{C}$, and the collection of types as the related category $\mathbf{C}^{ep}$ of embedding-projection pairs. What is then a dependent type F over D ? Since every preorder can be seen as a category it is natural to ask that F is a functor from D to $\mathbf{C}^{ep}$. Analogously a second order type will be seen as an endo-functor over $\mathbf{C}^{ep}$. However this will not suffice, for instance we will need that the family F preserves directed colimits, namely it is cocontinuous.

A problem of size arises when considering a second order type, as the domain (or base) of the family is a class, and not a set. This is precisely where the 'algebroidal' structure of $\mathbf{C}^{ep}$ may play an important role (see chpt. 5). It is then possible to determine a set of compact objects such that the behavior of a

cocontinuous functor is determined, up-to-isomorphism, by its behavior on these compact objects.

We can now summarize the contents of this chapter as follows. In section 1 we start by describing the constructions of sum and product in the category of sets. Next we give a rather general categorical treatment: given a family F as a functor F: $\mathbf{C}\to\mathbf{Cat}$ the Grothendieck construction $\mathbf{\Sigma_C F}$ will provide the interpretation of the sum. On the other hand $\mathbf{\Pi_C F}$ will be the category of "sections" of the "fibration" p: $\Sigma_C F\to\mathbf{C}$ that projects $\Sigma_C F$ onto $\mathbf{C}$. A section s of p has the property of being a functor s: $\mathbf{C}\to\Sigma_C F$ such that $p\circ s = id_C$.

Next we specialize these ideas to the categories of cpos and Scott domains.[13] The problem is to determine suitable continuity conditions so that the constructions of sum and product return domains. It turns out that everything works smoothly for dependent types. On the other hand second order types give some problems: (a) the sum $\Sigma_T F$ is not in general a domain; (b) the product $\Pi_T F$ is only *equivalent*, as a category, to a domain; (c) bifinite domains are not closed under the product $\Pi_T F$ (this motivates our shift towards Scott domains). At the end of the section we also sketch a model based on finitary projections where all the category theoretic constructions (e.g. $\Sigma_T F$) can be expressed in the framework of domain theory.

In section 2 we introduce a calculus of dependent and second order types and in section 3 we define its interpretation. This interpretation refers to the specific structure we have built. A general categorical treatment requires an amount of category theoretic background that goes beyond our goals, we address to the literature the interested reader (see, e.g., Jacobs&al.[91], Asperti&Longo[91]). Another interesting aspect that is omitted is the proof-theoretic analysis of the type system. Because of the size of the system and the inter-dependence of the judgments this is by no means a trivial task.

1. Domain Theoretic Constructions

We give the basic intuitions of the constructors Σ and Π in **Set**, we then abstract to **Cat**, and finally we specialize to certain categories of domains.

Dependent Types in Set

In set theory we may represent a family of sets as a function F: X→Set. More precisely we will consider a graph given as $\{(x, Fx)\}_{x\in X}$. We now formulate some basic constructions that will be suitably abstracted in the sequel.

In the first place one may build the (disjoint) *sum* of the sets in the family as:

$$\Sigma_X F \triangleq \{(x, y) \mid x\in X, y\in Fx\}$$

Observe that there is a projection map p: $\Sigma_X F\to X$ that is defined as $p(x, y) \triangleq x$. On the other hand one may build a *product* of the sets in the family as:

[13] The existence of least elements in domains is not a relevant hypothesis in this chapter.

$\Pi_X F \triangleq \{f: X \to \cup_{x \in X} Fx \mid \forall x \in X. fx \in Fx\}$

There is another way to write $\Pi_X F$ using the notion of "section" of the projection map $p: \Sigma_X F \to X$. A section is a map $s: X \to \Sigma_X F$ such that $p \circ s = id_X$, in other words for any $x \in X$ the section s picks up an element y in Fx. It is then clear that the collection of sections of p is in bijective correspondence with $\Pi_X F$.

Exercise ΣΠ-Set: Verify that the definitions of $\Sigma_X F$ and $\Pi_X F$ can be completed so to obtain sum and product of the objects in the family in the category of sets.

Exercise ΣΠ-Const: Suppose that the family $F: X \to Set$ is constant, say $F(x) = Y$ for each x in X. Then verify that $\Sigma_X F \cong X \times Y$, and $\Pi_X F \cong X \to Y$.

Exercise (P. Freyd) SmallCompl ⇒ Preorder: Show that every small category with arbitrary products is a poset. We recall that the category is small if the collection of its arrows is a set. *Hint:* Given two maps f, g: a→b in the small complete category **C** consider $\prod_I b$. The cardinality of $\mathbf{C}[a, \prod_I b]$ exceeds that of $Arr_{\mathbf{C}}$ when I is big enough.

Remark *(on the size of the family)*

Observe that in the definition of $\Sigma_X F$ and $\Pi_X F$ it is important that X is a set, so that the graph of F is again a set, and so are $\Sigma_X F$ and $\Pi_X F$. This observation preludes the problem we will find when dealing with second order types. In the interpretation suggested above neither the graph of a family F: Set→Set nor $\Sigma_{Set} F$ and $\Pi_{Set} F$ turn out to be sets !

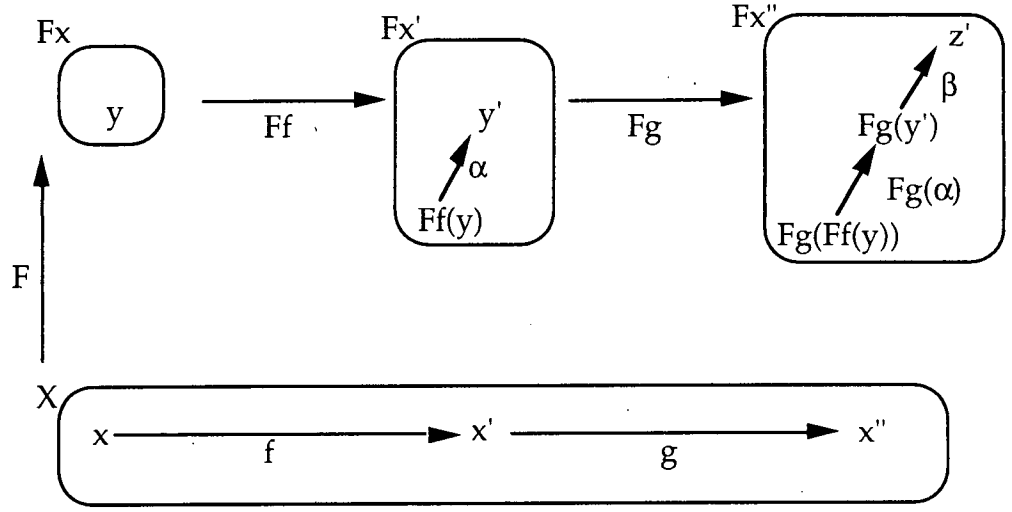
Dependent Types in Cat

We now abstract the previous constructions to categories. Let $F: \mathbf{X} \to \mathbf{Cat}$ be a functor where **X** is a small category, we define the categories $\Sigma_X F$, $\Pi_X F$, and the functors $p: \Sigma_X F \to \mathbf{X}$,[14] and $s: \mathbf{X} \to \Sigma_X F$ as follows:

$\Sigma_X F \triangleq \{(x, y) \mid x \in X, y \in Fx\}$
$\Sigma_X F\,[(x, y), (x', y')] \triangleq \{(f, \alpha) \mid f: x \to x',\ \alpha: F(f)(y) \to y'\}$
$id_{(x, y)} \triangleq (id_x, id_y)$
$(g, \beta) \circ (f, \alpha) \triangleq (g \circ f, \beta \circ (Fg)(\alpha))$

Here is the diagram that describes composition in $\Sigma_X F$:

[14] This projection is sometime referred to as Grothendieck fibration.



The functor p: $\Sigma_{\mathbf{X}}F \to \mathbf{X}$ is defined as:

$p(x, y) \triangleq x$

$p(f, \alpha) \triangleq f$

The category $\Pi_{\mathbf{X}}F$ is defined as:

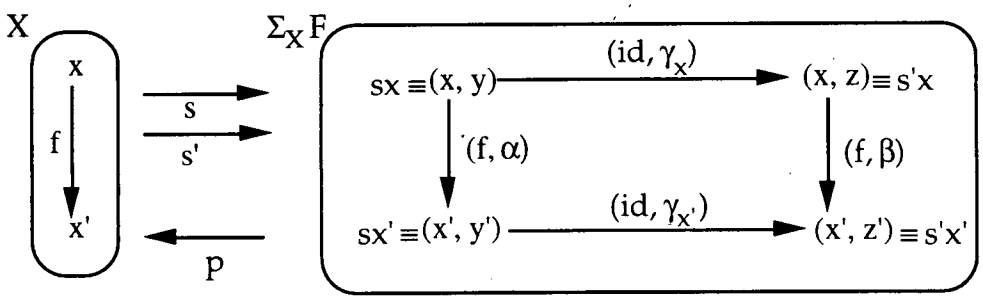
$\Pi_{\mathbf{X}}F \triangleq \{s: \mathbf{X} \to \Sigma_{\mathbf{X}}F \mid p \circ s = id_{\mathbf{X}}\}$

$\Pi_{\mathbf{X}}F[s, s'] \triangleq \{\nu: s \to s' \mid \nu$ is a "cartesian" natural transformation$\}$.

Observe that for a section we have:

$x \in \mathbf{X} \Rightarrow s(x) = (x, y)$ and $f \in Arr_{\mathbf{X}} \Rightarrow s(f) = (f, \alpha)$

A *cartesian* natural transformation $\nu: s \to s'$ is determined by a family $\{\nu_x \equiv (id_x, \gamma_x)\}_{x \in \mathbf{X}}$ where: $s(x) = (x, y)$, $s'(x) = (x, z)$, $\gamma_x: y \to z$, so the first component of the natural transformation is constrained to be the identity. Here is a diagram of the situation:



Dependent Types in Cpo

We are now going to refine this construction to the case in which we have a functor F: D→**Cpo**^{ep}, where D is a cpo and **Cpo**^{ep} is the category of cpos and embedding-projection pairs (i.e. X becomes a poset category D and the codomain of the functor is **Cpo**^{ep}). In order to have good closure properties it will be necessary to make some cocontinuity assumptions on the functor F as well as on the sections of the Grothendieck fibration.

Conventions

If $d \leq d'$ in D then we also denote with $d \leq d'$ the unique map from d to d' in the poset category D. If $f: D \to E$ is a morphism in $\mathbf{Cpo}^{ep}$ then we denote with f^+ the embedding and with f^- the projection.

Proposition *(Dependent Sum in Cpo^{ep})*

Let D be a cpo and $F: D \to \mathbf{Cpo}^{ep}$ be a functor, then

$$\Sigma_D F \triangleq \{(d, e) \mid d \in D, e \in Fd\}, \text{ ordered by}$$
$$(d, e) \leq_\Sigma (d', e') \Leftrightarrow d \leq_D d' \wedge F(d \leq d')^+(e) \leq_{F(d')} e'$$

is a cpo (and an instance of the general categorical construction).

Proof

Observe that $\mathbf{Cpo}^{ep}$ is the same as the category $\mathbf{Cpo}^{e}$ (with embeddings only as morphisms) which is a subcategory of **Cat**. It is immediate to verify that $(\Sigma_D F, \leq_\Sigma)$ is a poset with least element $(\perp_D, \perp_{F(\perp_D)})$. Verify: $(d, e) \leq_\Sigma (d', e')$ iff $\#\Sigma_D F[(d,e), (d',e')] = 1$.

Next let $X \equiv \{(d_i, e_i)\}_{i \in I}$ be directed in $\Sigma_D F$. Set for $d = \sqcup_{i \in I}\{d_i\}$,

$$\sqcup X = (d, \sqcup_{i \in I} F(d_i \leq d)^+(e_i))$$

We claim that this is well defined and the lub of X in $\Sigma_D F$.

- $\{F(d_i \leq d)^+ e_i\}_{i \in I}$ is directed. Since X is directed:
  $\forall i, j. \exists k. (d_i \leq d_k \wedge d_j \leq d_k \wedge F(d_i \leq d_k)^+(e_i) \leq e_k \wedge F(d_j \leq d_k)^+(e_j) \leq e_k)$. Hence:
  $F(d_i \leq d)^+(e_i) = F(d_k \leq d)^+ \circ F(d_i \leq d_k)^+(e_i) \leq F(d_k \leq d)^+(e_k)$, and similarly for j.
- $\sqcup X$ is an upper bound for X. Immediate, by definition.
- $\sqcup X$ is the lub. If (d',e') is an ub for X then it is clear that $d \leq d'$. Next observe:
  $F(d \leq d')^+(\sqcup_{i \in I} F(d_i \leq d)^+(e_i)) = \sqcup_{i \in I} F(d \leq d')^+(F(d_i \leq d)^+(e_i)) = \sqcup_{i \in I}(F(d_i \leq d')^+(e_i)) \leq e'$. □

Proposition *(Dependent Product in Cpo^{ep})*

Let D be a cpo and $F: D \to \mathbf{Cpo}^{ep}$ be a functor, then

$$\Pi_D F \triangleq \{s: D \to \Sigma_D F \mid s \text{ continuous}, p \circ s = id_D\}$$

with the order induced by $D \to \Sigma_D F$ is a cpo.

Proof

First observe that $p: \Sigma_D F \to D$ is continuous as for any $\{(d_i, e_i)\}_{i \in I}$ directed set in $\Sigma_D F$ we have: $p(\sqcup_{i \in I}(d_i, e_i)) = p(\sqcup_{i \in I} d_i, \sqcup_{i \in I} F(d_i \leq d)^+(e_i)) = \sqcup_{i \in I} d_i = \sqcup_{i \in I} p(d_i, e_i)$.

We also can define a least section as: $s(d) = (d, \perp_{F(d)})$.

Next observe that for any $\{s_i\}_{i \in I}$ directed set in $\Pi_D F$ we have, for any $d \in D$:

$$p \circ (\sqcup_{i \in I} s_i)(d) = p(\sqcup_{i \in I} s_i(d)) = \sqcup_{i \in I} p(s_i(d)) = d .$$

Hence the lub of a directed sets of sections exists and is the same as the lub in $D \to \Sigma_D F$. □

Exercises: (1) Verify that the definition of $\Sigma_D F$ is an instance of the definition in **Cat**.

(2) Let D be a cpo and $F: D \to \mathbf{Cpo}^{ep}$ a functor. Show that $\Pi_D F$ is isomorphic to:

$\{f: D \to \cup_{d \in D} Fd \mid fd \in Fd,\ f \text{ is 'cocontinuous'}\}$, ordered by

$f \leq g \Leftrightarrow \forall d \in D.\ fd \leq_{Fd} fd'$

where we say that f is cocontinuous if:

(i) $F(d \leq d')^+(fd) \leq fd'$.

(ii) For any $\{d_i\}_{i \in I}$ directed in D s.t. $\sqcup_{i \in I} d_i = d$: $f(d) = \sqcup_{i \in I} F(d_i \leq d)^+(fd_i)$.

(3) Verify that the definition of $\Pi_D F$ in **Cpo** corresponds to select a full subcategory of 'cocontinuous' sections out of the general construction described for **Cat**. □

Dependent Types in S-domains

We denote with **S** (for Scott) the category of algebraic, bounded complete, cpos.[15] If we want that the constructions defined above return S-domains then it is enough to make the following assumptions: (i) the domain of the family is an S-domain, (ii) the codomain is the category $\mathbf{S}^{ep}$ of S-domains and embedding-projection pairs, and, less obviously, (iii) the functor F is 'cocontinuous' in a sense which we define.

Definition *(directed colimits and cocontinuous functor)*

A *directed diagram* is a diagram indexed over a directed set. We say that a category has *directed colimits* if it has colimits of directed diagrams. We say that a functor is *cocontinuous* if it preserves colimits of directed diagrams.

Applying the theory developed in chpt. 5, it is easy to derive the following

Facts

(1) The category $\mathbf{S}^{ep}$ has directed colimits.

(2) Given a S-domain D and a functor $F: D \to \mathbf{S}^{ep}$, F is cocontinuous iff for any $\{d_i\}_{i \in I}$ directed in D s.t. $\sqcup_{i \in I} d_i = d$: $\sqcup_{i \in I} F(d_i \leq d)^+ \circ F(d_i \leq d)^- = id_D$.

(3) A functor $F: \mathbf{S}^{ep} \to \mathbf{S}^{ep}$ is cocontinuous iff for any D S-domain and any $\{p_i\}_{i \in I}$ directed set of projections over D $\sqcup_{i \in I} p_i = id_D \Rightarrow \sqcup_{i \in I} F(p_i) = id_{F(D)}$. □

Proposition *(Dependent Sum and Products in S-domains)*

Let D be an S-domain and $F: D \to \mathbf{S}^{ep}$ be a cocontinuous functor, then the cpos $\Sigma_D F$ and $\Pi_D F$ are S-domains.

Proof

• $\Sigma_D F$ is bounded complete. Let $X \equiv \{(d_i, e_i)\}_{i \in I}$ be bounded in $\Sigma_D F$ by (d', e'). Then (i) $\{d_i\}_{i \in I}$ is bounded in D by d' and therefore $\exists \sqcup_{i \in I} p_i \triangleq d$. (ii) Also $\{(F(d_i \leq d)^+(e_i)\}_{i \in I}$ is bounded by $F(d \leq d')^-(e')$ as:

$F(d_i \leq d')^+(e_i) = F(d \leq d')^+ F(d_i \leq d)^+(e_i) \leq e' \Rightarrow F(d_i \leq d)^+(e_i) \leq F(d \leq d')^-(e')$.

[15] It is possible to develop the theory by adding a condition on the countability of the compact elements.

Hence we set: $e \triangleq \bigsqcup_{i\in I} F(d_i \le d)^+(e_i)$. It is immediate to check that (d, e) is the lub.

• $\Sigma_D F$ is algebraic. We claim: (i) $\Sigma_D F_o = \{(d, e) \mid d \in D_o \wedge e \in F(d)_o\}$. (ii) for any $(d', e') \in \Sigma_D F$, $\downarrow(d', e') \cap \Sigma_D F_o$ is directed with lub (d', e').

(i) Let $X \equiv \{(d_i, e_i)\}_{i\in I}$ be directed in $\Sigma_D F$ with

(a) $d' \le \bigsqcup_{i\in I} d_i = d$, (b) $F(d' \le d)^+(e') \le \bigsqcup_{i\in I} F(d_i \le d)^+(e_i)$.

By hyp. d' and e' are compact, $F(d' \le d)^+(e')$ is also compact hence we can find j such that: $d' \le d_j$, $F(d' \le d)^+(e') \le F(d_j \le d)^+(e_j)$ that implies $F(d' \le d_j)^+(e') \le e_j$. That is $(d', e') \le (d_j, e_j)$.

(ii) The set is directed because $\Sigma_D F$ is bounded complete. Given (d, e) we consider:

- $\{d_i\}_{i\in I} \subseteq D_o$ directed s.t. $\bigsqcup_{i\in I} d_i = d$.
- for any $i \in I$ $\{e_{ij}\}_{j\in J_i} \subseteq F(d_i)_o$ directed s.t. $\bigsqcup_{j\in J_i} e_{ij} = F(d_i \le d)^-(e)$. Then:

$\bigsqcup_{i\in I\, j\in J_i}(d_i, e_{ij}) = (d, \bigsqcup_{i\in I\, j\in J_i} F(d_i \le d)^+(e_{ij})) = (d, \bigsqcup_{i\in I} F(d_i \le d)^+(\bigsqcup_{j\in J_i} e_{ij})) =$
$(d, \bigsqcup_{i\in I} F(d_i \le d)^+ F(d_i \le d)^-(e)) = (d, e)$, the latter by cocontinuity of F.

• $\Pi_D F$ is bounded complete. Suppose $\{s_i\}_{i\in I}$ is a bounded set in $\Pi_D F$. Since bounded completeness is preserved exponentiation we can compute $\bigsqcup_{i\in I} s_i$ in $D \to \Sigma_D F$. It remains to show that: $p \circ (\bigsqcup_{i\in I} s_i) = id_D$. Observe, any $d \in D$,

$p \circ (\bigsqcup_{i\in I} s_i)(d) = p(\bigsqcup_{i\in I} s_i(d)) = p(\bigsqcup_{i\in I}(d, e_i)) = d$.

• $\Pi_D F$ is algebraic. Consider the sections [d, e] for $d \in D_o$, $e \in F(d)_o$, defined as:

$[d, e](x) =$ if $d \le x$ then $(x, F(d \le x)^+(e))$ else $(x, \perp_{F(x)})$.

Verify that [d, e] is compact in $\Pi_D F$. It remains to observe that for any $s \in \Pi_D F$ $\{[d, e] \mid [d, e] \le s\}$ determines s. □

Second order types in S-domains

We now look for an interpretation of second order types as cpos and/or domains. Suppose that F: $\mathbf{S}^{ep} \to \mathbf{S}^{ep}$ is a cocontinuous functor. Then, as an instance of the general categorical constructions, we can form the categories $\Sigma_{\mathbf{S}^{ep}} F$ and $\Pi_{\mathbf{S}^{ep}} F$. In the first place we observe that the Σ-construction does not need to return a preorder as there can be several embedding projection pairs between two domains. We therefore concentrate our efforts on the Π-construction.

Given a cocontinuous functor F: $\mathbf{S}^{ep} \to \mathbf{S}^{ep}$ we restrict our attention to the *cocontinuous sections* s: $\mathbf{S}^{ep} \to \Sigma_{\mathbf{S}^{ep}} F$. A cocontinuous section s is a section characterized by the following property: for any D S-domain, $\{p_i\}_{i\in I}$ directed family of projections: $\bigsqcup_{i\in I} p_i = id_D \Rightarrow s(D) = \bigsqcup_{i\in I} F(p_i)^+(s(p_i(D)))$.

The first obvious problem is that the functor F as well as the cocontinuous sections are not sets as their description requires functions whose domain is the class of all cpos.

Suppose that **C** is an algebroidal category of domains and let $\{C_i\}_{i\in\omega}$ be an enumeration of the compact objects up to isomorphism (verify that this enumeration exists for the category $\mathbf{S}^{ep}$). Then the behaviour of a cocontinuous functor F can be described up to isomorphism as $\{(i, j) \mid C_j \sqsubseteq FC_i\}$. Moreover the behaviour of a cocontinuous section is completely determined by F and by the behaviour of the section on $\{C_i\}_{i\in\omega}$. Clearly we are mimicking the definition of the graph of a continuous function over algebraic cpos. The difficulty here lies in the

fact that colimits are not unique (up to equality). These considerations introduce the following theorem due to Coquand&al.[89], after Girard[86].

Theorem *(Second order product in S-domains)*

Let F: $\mathbf{S}^{ep}\to\mathbf{S}^{ep}$ be a cocontinuous functor then the poset of cocontinuous sections, that we denote with $\Pi_{\mathbf{S}^{ep}}F$, is equivalent, as a category, to an S-domain.

Proof

Let S_o be a countable enumeration, up to isomorphism, of compact objects in $\mathbf{S}^{ep}$. Now consider

$$P = \{f: S_o\to \cup_{D\in S_o}F(D) \mid f(D)\in F(D) \wedge (f: D\to E \text{ in } S_o{}^{ep} \Rightarrow F(f)^+(fD)\le fE)\},$$

with the pointwise order. One shows that P is an S-domain, equivalent as a category to $\Pi_{\mathbf{S}^{ep}}F$. Here we only describe the equivalence. It is clear that any cocontinuous section determines a family in P, by restriction to $S_o{}^{ep}$. Vice versa given a family f in P we define its extension to a cocontinuous section, (ext f), as:

$$(\text{ext } f)(E) = (E, \sqcup\{(Ff)^+(fD) \mid D\in S_o \wedge f: D\to E \text{ in } \mathbf{S}^{ep}\}). \qquad \square$$

In the following interpretation we will assume that $\Pi_{\mathbf{S}^{ep}}F$ denotes a canonically chosen S-domain.

Exercise Π(Id): Consider the identity functor Id: $\mathbf{S}^{ep}\to\mathbf{S}^{ep}$. Prove that $\Pi_{\mathbf{S}^{ep}}Id$ is the cpo with one element. *Hint:* let f be a continuous section and D an S-domain. Then there are two standard embeddings, inl and inr, of D in D+D, where + is the coalesced sum. The condition on sections requires that $f(D)\le f(D+D)$, but this forces $F(D) = \bot_D$.

Notes

(1) The previous exercise hints at the fact that cocontinuous sections satisfy certain 'uniformity conditions', namely the choice of the element has to be invariant w.r.t. to certain embeddings. In practice the syntax suggests that all the definable functors are very uniform so one can look for even stronger uniformity conditions in the model. Here is one that arises in the 'stable case' (see chpt. 7, Girard[86]) and that leads to a "smaller" interpretation of certain types. Every section s of p: $\Sigma_{\mathbf{S}^{ep}}F\to\mathbf{S}^{ep}$ has to satisfy the condition:

$$h: D\to E \text{ in } \mathbf{S}^{ep} \Rightarrow sD = (Fh)^-(sE)$$

Note that this condition implies the standard condition in the continuous case.

(2) It can be proved that bifinite domains are not closed w.r.t. the $\Pi_T F$ construction. The basic problem arises from the observation that $(\mathbf{S}^{ep})_o$ does not need to satisfy property M (see Jung[91]).

Finitary Projections Model

In chpt. 5 we have discussed how to represent (countably based) S-domains as finitary projections over a universal domain U. In this section we briefly describe

the construction of the operators Σ and Π in this framework (see Amadio&al[86]).

Suppose that U is an S-domain such that:

$U\times U \trianglelefteq U$ via $(\lambda(u, u').\ \langle u, u'\rangle, \lambda u.\ ((\mathrm{fst}\ u), (\mathrm{snd}\ u)))$: $U\times U\to U$, and
$(U\to U) \trianglelefteq U$ via (i, j): $(U\to U)\to U$.

We also know that (see chapter 5):

$FP(U) \trianglelefteq (U\to U)$ via $(id_{FP(U)}, \pi)$.

We set: $\underline{\pi} = i\circ\pi\circ j$. We now suppose the following correspondences:

- the projection $p\in FP(U)$ represents the domain im(p).
- a function f: $U\to U$ such that $f = \underline{\pi}\circ f\circ p$ represents a (cocontinuous) functor from the domain im(p) to the domain of types FP(U).

It is important to realize that FP(U) and $\mathbf{S}^{ep}$ (well, the subcategory of countably based domains) are not equivalent categories as FP(U) is just a poset. As a matter of fact we will get a different model where, in particular, one can interpret $\Sigma_T F$, the second order Σ-construction, as a domain.

Definition *(Σ and Π constructions in FP(U))*

Let $p\in FP(U)$, and f: $U\to U$ be such that $f = \underline{\pi}\circ f\circ p$. We define:

$\Sigma_p f \triangleq \lambda u.\langle p(\mathrm{fst}\ u), (f(\mathrm{fst}\ u))(\mathrm{snd}\ u)\rangle$: $U\to U$,
$\Pi_p f \triangleq \lambda u.i(\lambda x.j(fx)((ju)(pt))$: $U\to U$.

The following proposition establishes that $\Sigma_p f$ and $\Pi_p f$ are finitary projections representing the 'right' domains. Here $\lambda d\in im(p).\ im(f\ d)$ should be thought as a functor (actually a continuous function) from im(p) to FP(U).

Proposition

Let $p\in FP(U)$, and f: $U\to U$ be such that $f = \underline{\pi}\circ f\circ p$. Then:

(1) $\Sigma_p f, \Pi_p f \in FP(U)$.
(2) $im(\Pi_p f) \cong \Pi_{im(p)}\ (\lambda d\in im(p).\ im(f\ d))$,
$im(\Sigma_p f) \cong \Sigma_{im(p)}\ (\lambda d\in im(p).\ im(f\ d))$.

Exercise FP.Π(Id): Compute Π_πId. Compare the corresponding domain with the one that is obtained in ex. Π(Id).

2. A Calculus of Dependent and Second Order Types

In this section we introduce the typing rules of a typed lambda calculus with dependent and second order types.

Syntactic Categories

The syntactic categories are presented in BNF. In specifying the rules we will indicate types with A, B,..., and terms with M, N,...

Variables	$V ::= x \mid y \mid \ldots$
Contexts	$\Gamma ::= \varnothing \mid \Gamma, (V: A) \mid \Gamma, (V: tp)$
Types[16]	$A ::= V \mid (\Pi V: A.\ A) \mid (\Pi V: tp.\ A) \mid (\Sigma V: A.\ A)$
Terms	$M ::= V \mid (\lambda V: A.\ M) \mid (\lambda V: tp.\ M) \mid (MM) \mid (MA) \mid$ $\langle M, M\rangle \mid (fst\ M) \mid (snd\ M)$

Judgments

Well Formed Context	Γ ok
Well Formed Type	$\Gamma \supset A: tp$
Well Formed Term	$\Gamma \supset M: A$

Well Formed Contexts

(Context.Empty)	$\Rightarrow \varnothing$ ok
(Context.Intro.Type)	Γ ok , $x \notin Dom(\Gamma) \Rightarrow \Gamma, (x: tp)$ ok
(Context.Intro.Term)	$\Gamma \supset A: tp,\ x \notin Dom(\Gamma) \Rightarrow \Gamma, (x: A)$ ok

Well Formed Types

(Type.Assmp)	$(x: tp) \in \Gamma, \Gamma$ ok $\Rightarrow \Gamma \supset x: tp$
(Type.Π.Depend)	$\Gamma, (x: A) \supset B: tp \Rightarrow \Gamma \supset (\Pi x: A.B): tp$
(Type.Σ.Depend)	$\Gamma, (x: A) \supset B: tp \Rightarrow \Gamma \supset (\Sigma x: A.B): tp$
(Type.Π.IIord)	$\Gamma, (x: tp) \supset B: tp \Rightarrow \Gamma \supset (\Pi x: tp.B): tp$

Well Formed Terms

(Term.Assmp)	$(x: A) \in \Gamma, \Gamma$ ok $\Rightarrow \Gamma \supset x: A$
(Term.Π.Depend.Intro)	$\Gamma, (x: A) \supset M: B \Rightarrow \Gamma \supset (\lambda x: A.M): (\Pi x: A.B)$
(Term.Π.Depend.Elim)	$\Gamma \supset M: (\Pi x: A.B), \Gamma \supset N: A \Rightarrow \Gamma \supset (MN): [N/x]\ B$
(Term.Σ.Depend.Intro)	$\Gamma \supset M: A, \Gamma, (x: A) \supset N: B \Rightarrow \Gamma \supset \langle M, [M/x]N\rangle: (\Sigma x: A.B)$
(Term.Σ.Depend.Elim1)	$\Gamma \supset M: (\Sigma x: A.B) \Rightarrow \Gamma \supset (fst\ M): A$
(Term.Σ.Depend.Elim2)	$\Gamma \supset M: (\Sigma x: A.B) \Rightarrow \Gamma \supset (snd\ M): [(fst\ M)/x]\ B$
(Term.Π.IIord.Intro)	$\Gamma, (x: tp) \supset M: B \Rightarrow \Gamma \supset M: (\Pi x: tp.B)$
(Term.Π.IIord.Elim)	$\Gamma \supset M: (\Pi x: tp.B), \Gamma \supset A: tp \Rightarrow \Gamma \supset (MA): [A/x]\ B$

Note: In this presentation of the system we have not introduced equality judgments for types and terms as they are not needed in order to give an idea of the interpretation. However this gives a deceptive impression of simplicity. A complete work should include the description of the equality rules and the verification of their sound interpretation.

[16] In order to write real, closed dependent types in this calculus one has to assume the existence of some constant dependent type.

3. Interpretation

In first approximation the interpretation of a context judgment, Γ ok, is a category, the interpretation of tp is the category $\mathbf{S}^{ep}$, the interpretation of a type judgment, $\Gamma \supset A$: tp, is a functor from the interpretation of Γ to the interpretation of tp, and the interpretation of a term judgment, $\Gamma \supset M$: A, is a section of the Grothendieck fibration p: $\Sigma_{[\![\Gamma]\!]} F \to [\![\Gamma]\!]$, where F is the functor given by the interpretation of $\Gamma \supset A$: tp. Note that the interpretations are inter-dependent, their well-foundation can only be obtained by a careful examination of the syntax. In the following we define the interpretation by induction on the derivation of the judgment.

Proviso

When defining the interpretation by induction on the length of the proof we may need certain assumptions that have been omitted, for the sake of brevity, in the presentation of the system. For example if we interpret $\Gamma \supset (\lambda x\text{: } A.M)\text{: } (\Pi x\text{: } A.B)$ then we need the interpretation of $\Gamma, (x\text{: } A) \supset M\text{: } B$ but also the one of $\Gamma \supset A$: tp. In order to give a rigorous treatment one should either make the system a bit heavier or show that there is always a *canonical* way to select a proof of, say, $\Gamma \supset A$: tp, from a proof of $\Gamma, (x\text{: } A) \supset M\text{: } B$. Also note that the interpretation is presented in a set theoretic style, of course one should make sure that the set-theoretic transformations we use exist in the domain theoretic model !

Context Interpretation

The interpretation of a context is a category. We start with the trivial category 1, and we use the category $\mathbf{S}^{ep}$ to interpret tp. Next we build new categories using the Grothendieck construction.

$[\![\varnothing \text{ ok}]\!] = 1$

$[\![\Gamma, (x\text{: tp}) \text{ ok}]\!] = [\![\Gamma \text{ ok}]\!] \times \mathbf{S}^{ep}$

$[\![\Gamma, (x\text{: A}) \text{ ok}]\!] = \Sigma_{[\![\Gamma]\!]} [\![\Gamma \supset A\text{: tp}]\!]$

Type Interpretation

The interpretation of a type judgment is a functor into the category $\mathbf{S}^{ep}$. Given a variable, say x, occurring in the well formed context Γ there is some standard way to define a selector π_x, as composition of projections, that returns the component corresponding to x in the interpretation of Γ.

$[\![\Gamma \supset x\text{: tp}]\!] = \pi_x$

$[\![\Gamma \supset (\Pi x\text{: } A.B)\text{: tp}]\!] = \lambda y.\ \Pi_{Gy}\ (\lambda x.F(y,x))$

$[\![\Gamma \supset (\Sigma x\text{: } A.B)\text{: tp}]\!] = \lambda y.\ \Sigma_{Gy}\ (\lambda x.F(y,x))$

where $F = [\![\Gamma, (x\text{: } A) \supset B\text{: tp}]\!]$, and $G = [\![\Gamma \supset A\text{: tp}]\!]$

$[\![\Gamma \supset (\Pi x\text{: tp}.B)\text{: tp}]\!] = \lambda y.\ \Pi_{\mathbf{S}^{ep}}\ (\lambda x.F(y,x))$, where $F = [\![\Gamma, (x\text{: tp}) \supset B\text{: tp}]\!]$

Term Interpretation

As mentioned above the interpretation of a term judgment $\Gamma \supset M: A$ is a section of the fibration $p: \Sigma_{[\![\Gamma]\!]} F \to [\![\Gamma]\!]$, where F is the functor given by the interpretation of the typing judgement $\Gamma \supset A: tp$.

$[\![\Gamma \supset x: A]\!] = \pi_x$

$[\![\Gamma \supset (\lambda x: A.M): (\Pi x: A.B)]\!] = \lambda y \in [\![\Gamma]\!].\ \lambda x \in Gy.\ ([\![\Gamma, (x: A) \supset M: B]\!])(y, x)$
where $G = [\![\Gamma \supset A: tp]\!]$

$[\![\Gamma \supset (MN): [N/x]B]\!] = \lambda y \in [\![\Gamma]\!].\ ([\![\Gamma \supset M: (\Pi x: A.B)]\!]\ y)\ ([\![\Gamma \supset N: A]\!]\ y)$

$[\![\Gamma \supset \langle M, N\rangle: (\Sigma x: A.B)]\!] = \lambda y \in [\![\Gamma]\!].\ (ay, (by)(ay))$
where $a = [\![\Gamma \supset M: A]\!]$, $b = [\![\Gamma, (x: A) \supset N: B]\!]$

$[\![\Gamma \supset (\text{fst } M): A]\!] = \lambda y \in [\![\Gamma]\!].\ \pi_1([\![\Gamma \supset M: (\Sigma x: A.B)]\!]\ y)$

$[\![\Gamma \supset (\text{snd } M): [(\text{fst } M)/x]\ B]\!] = \lambda y \in [\![\Gamma]\!].\ \pi_2([\![\Gamma \supset M: (\Sigma x: A.B)]\!]\ y)$

$[\![\Gamma \supset M: (\Pi x: tp.B)]\!] = \lambda y \in [\![\Gamma]\!].\ \lambda x \in \mathbf{S}^{ep}.\ ([\![\Gamma, (x: tp) \supset M: B]\!])(y, x)$

$[\![\Gamma \supset (MA): [A/x]B]\!] = \lambda y \in [\![\Gamma]\!].\ ([\![\ \Gamma \supset M: (\Pi x: tp.B)]\!]\ y)\ ([\![\Gamma \supset A: tp]\!]\ y)$

Exercise tp:tp : Propose an interpretation in the finitary projection model for the calculus extended with a rule that makes *tp* into a type:

(tp:tp) $\quad \Gamma\ ok \Rightarrow \Gamma \supset tp: tp$

Hint: the finitary projection π represents the collection of all types (see Amadio&al[86]).

Guide to the References: In Coquand&al.[91] one will find the details of the domain-theoretic interpretation of system F. Asperti&Longo[91] contains a rather complete analysis of the categorical structure needed to interpret system F from the viewpoint of 'indexed category theory' and 'internal category theory'. There have been several approaches to the problem of describing what is a categorical model of dependent types, second order types, and more. See Jacobs&al.[91] for an up-to-date account using fibrations.

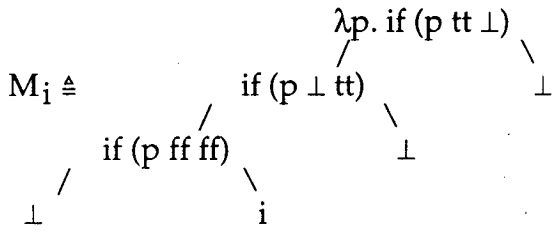
7. Domains and Stable Maps

Contents: 1. Stable Bifinites, 2. A Characterization of Stable Bifinites, 3. Traces of Stable Functions, 4. Some Domain Theoretic Constructions, Appendix 1: Functional Spaces and ω-algebraicity, Appendix 2: L-domains in the Stable Case.

The notion of *stable map*, as that of a continuous map that preserves meets of pairs of compatible elements, has two origins related to λ-calculus and proof theory. It was introduced by Berry as a semantic approximation of the sequential behavior of λ-calculus evaluation[17] and by Girard in the theory of dilators. It appears that similar ideas were also developed independently and with purely algebraic motivations by Diers (see Taylor[90] for an up-to-date account oriented towards computer science).

As an initial motivation let us mention some facts about the 'full abstraction problem for PCF' (a simply typed lambda calculus with a fixed point combinator and arithmetic functions, see chpt. 1). We know how to build a standard model of this calculus working in cartesian closed categories of algebraic cpos and continuous functions. It turns out that this model is 'adequate but not fully abstract'. This means that the model-theoretic pre-order on terms is strictly contained in the operational pre-order.

We describe Plotkin's counter-example. Define a family of terms M_i:(bool→bool→bool)→nat, representing the following function:



Observe that if $i \neq j$, M_i and M_j can be separated by a term p such that:

$p(tt, \bot) = tt \qquad p(\bot, tt) = tt \qquad p(ff, ff) = ff$

for example a parallel or, which is a compact element of the continuous model.

It turns out that a term that behaves like p is not definable in PCF as the evaluation of the arguments necessarily proceeds "sequentially". Therefore terms that are "operationally equivalent", like the M_i defined above, are not equated in the model based on the continuous functions. As a matter of fact R. Milner has shown that any "algebraic continuous model" in which operational and model equivalence coincide has the property that *all compact elements are definable* in

[17] The stable maps considered here are in Berry's terminology "multiplicative under condition".

the calculus.

Hence a possible strategy towards the goal of obtaining a fully abstract model is to build a model in which certain compact elements are ruled out. Observe, for example, that the function por does not preserve meets and thus it is *not* stable. Unfortunately the stable model does not provide the fully abstract model, it does provide, however, another theory of finite approximation that can stand a comparison with Scott theory and which has had interesting applications.

Once the notion of stable transformation is accepted and we start working with categories of cpos having meets of compatible elements two basic original ideas appear. The first one is a *new order on the functional space* that refines the pointwise order. This order is naturally discovered by requiring the stability of the evaluation. The second one is *property I*. It says that the principal ideal generated by a compact element is finite. This property has to do with the ω-algebraicity of the functional space. It also appears in Kahn and Plotkin's "concrete domains", and in Winskel's "event structures". It is worth observing that property I is *not* preserved by the pointwise ordering, so it also provides an a posteriori justification for the use of the stable ordering. More importantly under property I it is possible to recover the original *computational intuition* of stable map as that of a continuous map over domains, say f: D→E, with the property that:

$$\forall d, e.\ e \leq f(d) \Rightarrow \exists \min\{d' \leq d \mid e \leq f(d')\}.$$

Roughly, it is always possible to find "locally" the minimum amount of information needed to generate a given output.

In the following we are particularly indebted to Berry[79]. There will be however a notable variation: we will consider a "bifinite approach to the stable case" (Amadio[91]).[18] Berry worked with dI-domains, which are bounded complete ω-algebraic cpos satisfying property I and distributivity. We already discussed the reasons for the introduction of property I. *Distributivity* arises when trying to prove that the functional space is *bounded complete*. On the other hand in our approach we will drop the requirement of bounded completeness and substitute it with the weaker assumption that compatible elements have meets. Then it will turn out that the condition of distributivity can be dropped as well.[19]

An *overview* of this chapter goes as follows: in section 1 we introduce the ccc of stable bifinites and stable maps, $\mathrm{Bif}_{\wedge}$; in section 2 we characterize the objects in $\mathrm{Bif}_{\wedge}$; in section 3 we introduce a notion of *trace* which is useful in analysing the stable exponent; in section 4 we study some more closure properties of the category; we conclude with some observation on the properties necessary to preserve the ω-algebraicity of the functional space (A.1), and the construction of a ccc of 'stable' L-domains.

[18] As in chapter 5 in this chapter bifinite means ω-bifinite.

[19] In the following we will restrict our attention to domains with a least element, the more general theory can also be developed along the lines exposed in the continuous case.

1. Stable Bifinites

We consider the category $\mathbf{Cpo}_\wedge$ of cpos with continuous meets of pairs of compatible elements, and continuous maps preserving these meets. Sometimes we briefly refer to these cpos as *meet cpos*. We first show that $\mathbf{Cpo}_\wedge$ is a ccc. The main novelty w.r.t. the continuous case lies in the introduction of a *stable ordering* on the functional space that is practically forced by the requirement that the evaluation map is stable.

Definition *(meet cpos)*

$D \in Cpo_\wedge$ if (1) D is a cpo, (2) $\forall d, e \in D.(d \uparrow e \Rightarrow \exists\, d \wedge e)$,
(3) $\forall X \subseteq D$, directed $(d \uparrow \sqcup X \Rightarrow d \wedge \sqcup X = \sqcup_{x \in X}(d \wedge x))$.[20]

Cpos satisfying these conditions are called meet cpos.

Definition *(category of meet cpos and stable maps)*

Let D, E be meet cpos. A map f: D→E is stable if
(1) it is Scott continuous, and (2) $\forall d, e \in D.(d \uparrow e \Rightarrow f(d \wedge e) = f(d) \wedge f(e))$.
We denote with $\mathbf{Cpo}_\wedge$ the category of meet cpos and stable maps.

Proposition *($Cpo_\wedge$ is a ccc)*

The category $\mathbf{Cpo}_\wedge$ is cartesian closed. In particular given D, $E \in Cpo_\wedge$ the exponent D→E is given by the collection of stable maps ordered as follows:

$$f \preceq g \Leftrightarrow \forall d, d' \in D.(d \leq d' \Rightarrow f(d) = f(d') \wedge g(d)).^{21}$$

Proof

We assume that by now the reader is familiar with the proof that cpos and continuous maps form a ccc. We mainly refer to the additional checks that are needed.

$\mathbf{Cpo}_\wedge$ is *cartesian*: take the terminal object and the product as in **Cpo**. Check: (a) terminal object and product are in $Cpo_\wedge$. (b) constant functions and projections are stable. (c) if f: C→D and g: C→E are stable then the pairing $\langle f, g\rangle \triangleq \lambda c.(fc, gc)$: C→D×E is stable. Let us give the details of this. Suppose $c \uparrow c'$, then:

$$\langle f, g\rangle(c \wedge c') = (f(c \wedge c'), g(c \wedge c')) = (f(c) \wedge f(c'), g(c) \wedge g(c')) = (f(c), g(c)) \wedge (f(c'), g(c')).$$

$\mathbf{Cpo}_\wedge$ is *cartesian closed.* Consider the relation $\preceq$ on the collection of stable maps D→E defined as above. First let us control that $(D \to E, \preceq) \in Cpo_\wedge$.

- $(D \to E, \preceq)$ is a partial order with least element.

Reflexivity: $f \preceq f$. $d \leq d' \Rightarrow f(d) = f(d') \wedge f(d)$, by monotonicity.
Symmetry: $f \preceq g \preceq f \Rightarrow f = g$. $d \leq d \Rightarrow f(d) = f(d) \wedge g(d) \leq g(d)$ and symmetrically $d \leq d \Rightarrow g(d) = f(d) \wedge g(d) \leq f(d)$.

---

[20] Note that (3) makes sense. If D is algebraic then (2) implies (3). We use (3) many times in showing that D→E is a $Cpo_\wedge$. Another way to state (3) is to say that whenever we look at principal ideals the $\wedge$ distributes w.r.t. directed sets.

[21] Henceforth we denote with $\preceq$ the stable ordering on the functional space.

Transitivity: $f \preceq g \preceq h \Rightarrow f \preceq h$. $d \leq d' \Rightarrow f(d) = f(d') \wedge g(d), g(d) = g(d') \wedge h(d) \Rightarrow f(d) = f(d') \wedge g(d') \wedge h(d) = f(d') \wedge h(d)$.
Least Element: $\lambda d.\perp \preceq f$ as $d \leq d' \Rightarrow \perp = \perp \wedge g(d)$.

• Compatible elements have a glb. Given $f, g \preceq h$ define $(f \wedge g)(d) \triangleq f(d) \wedge g(d)$. Let us verify $f \wedge g$ is well-defined and stable. Well defined: observe $f(d), g(d) \leq h(d)$. Let X directed in D. Then: $(f \wedge g)(\sqcup X) = f(\sqcup X) \wedge g(\sqcup X) = \sqcup f(X) \wedge \sqcup g(X) = \sqcup_{x \in X} (f(x) \wedge g(x))$, by the assumption that $\wedge$ is continuous. Moreover: $\sqcup_{x \in X} (f(x) \wedge g(x)) = \sqcup_{x \in X} (f \wedge g(x))$. Suppose $d \uparrow d'$. Then:

$$(f \wedge g)(d \wedge d') = f(d \wedge d') \wedge g(d \wedge d') = f(d) \wedge f(d') \wedge g(d) \wedge g(d') = (f \wedge g)(d) \wedge (f \wedge g)(d').$$

• $f \wedge g = \text{glb}\{f, g\}$. Recall: $f, g \preceq h$. Hence $d \leq d'$ implies $f(d) = f(d') \wedge h(d)$ and $g(d) = g(d') \wedge h(d)$. We first show $f \wedge g \preceq f$. $d \leq d' \Rightarrow f \wedge g(d) = f(d) \wedge g(d) = f(d') \wedge g(d') \wedge h(d) = f(d') \wedge g(d') \wedge f(d') \wedge h(d) = f \wedge g(d') \wedge f(d)$. And symmetrically $f \wedge g \preceq g$.
Suppose $k \preceq f, g$. This means: $d \leq d' \Rightarrow k(d) = k(d') \wedge f(d)$ and $k(d) = k(d') \wedge g(d)$. Now $k \preceq f \wedge g$ iff $d \leq d' \Rightarrow k(d) = k(d') \wedge f(d) \wedge g(d)$. Observe:

$$k(d) = k(d) \wedge k(d) = k(d') \wedge f(d) \wedge k(d') \wedge g(d) = k(d') \wedge f(d) \wedge g(d).$$

• Given F directed in $D \to E$ define $(\sqcup F)(d) \triangleq \sqcup_{f \in F} f(d)$. Check: $\sqcup F$ is well-defined and stable. Remark that if F is directed w.r.t. $\preceq$ then it is also directed w.r.t. the pointwise ordering. In particular $\{f(d)\}_{f \in F}$ is directed in E and therefore $(\sqcup F)$ is well-defined. $\sqcup F$ is a continuous map by the standard proof. Let us check stability: suppose $d \uparrow d'$. Then $(\sqcup F)(d \wedge d') = \sqcup_{f \in F} f(d \wedge d') = \sqcup_{f \in F} (f(d) \wedge f(d')) = \sqcup_{f \in F} f(d) \wedge \sqcup_{f \in F} f(d')$, since $\sqcup_{f \in F} f(d) \uparrow \sqcup_{f \in F} f(d')$ and $\wedge$ is continuous. Now:

$$\sqcup_{f \in F} f(d) \wedge \sqcup_{f \in F} f(d') = (\sqcup F)(d) \wedge (\sqcup F)(d').$$

• $\sqcup F = \text{lub } F$. First we show $f \in F \Rightarrow f \preceq \sqcup F$. Observe:
$\sqcup F(d) = \sqcup_{g \in F} g(d) = \sqcup\{g(d) \mid f \preceq g, g \in F\}$. Now $f \preceq g, d \leq d' \Rightarrow f(d) = f(d') \wedge g(d)$. So:

$$f(d) = \sqcup\{f(d') \wedge g(d) \mid f \preceq g, g \in F\} = f(d') \wedge \sqcup_{g \in F} g(d) = f(d') \wedge \sqcup F(d).$$

Suppose $\forall g \in F.(g \preceq h)$. Then $d \leq d' \Rightarrow g(d) = g(d') \wedge h(d)$. Hence:

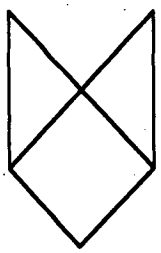
$$\sqcup F(d) = \sqcup_{g \in F} g(d) = \sqcup_{g \in F} g(d') \wedge h(d) = \sqcup F(d) \wedge h(d), \text{ and } \sqcup F \preceq h.$$

• $\wedge$ is continuous. Suppose $f \uparrow \sqcup F$, F directed. .
$(f \wedge \sqcup F)(d) = f(d) \wedge \sqcup_{g \in F} g(d) = \sqcup_{g \in F} f(d) \wedge g(d) = \sqcup_{g \in F} (f \wedge g)(d) = (\sqcup_{g \in F} (f \wedge g))(d)$.

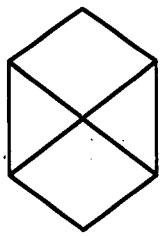
Finally verify that: $\text{eval}_{D,E} \triangleq \lambda f.\lambda d.f(d, e)$: $(D \to E) \times D \to E$ and $\Lambda_{D,D',E} \triangleq \lambda g.\lambda d'.\lambda e.g(d, e)$: $(D' \times E \to D) \to (D' \to (E \to D)$ are stable. □

Notes

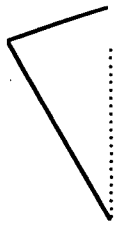
(1) Here are some examples to keep in mind:



$cpo_\wedge$, ¬bound. compl.



$\neg\ cpo_\wedge$



¬ω-algebraic, ¬cont. inf

(2) Where does the stable order come from? Suppose we want to show that $\mathbf{Cpo}_\wedge$ is cartesian closed. One problem is to find an adequate notion of order so that the functions of evaluation and currying turn out to be stable. Take the collection of stable maps with an unspecified order. Suppose f, g: D→E, d, d'∈D and (f, d)↑(g, d'). Then we expect: eval((f, d)∧(g, d')) = eval(f∧g, d∧d') = f∧g(d∧d') = f(d∧d')∧g(d∧d') = f(d)∧f(d')∧g(d)∧g(d'). On the other hand if eval is stable then: eval((f, d)∧(g, d')) = eval(f, d)∧eval(g, d') = f(d)∧g(d'). So the following has to hold: (f, d)↑(g, d') ⇒ f(d)∧f(d')∧g(d)∧g(d') = f(d)∧g(d'). If, for instance, we assume that d≤d', g is less than f and the "unknown order" includes the pointwise order we get: g(d) = f(d)∧g(d'), the stable order! One of the checks left to the reader in the proposition above is precisely the stability of eval w.r.t. the stable order of the functional space.

(3) Let D, E ∈$Cpo_\wedge$, f, g: D→E, f continuous, g stable and f ≤ g. Then f is stable. Suppose x, y ≤ z. Then:

$$f(x\wedge y) = f(z) \wedge g(x\wedge y) = f(z) \wedge g(x) \wedge g(y) = f(z) \wedge g(x) \wedge f(z) \wedge g(y) = f(x)\wedge f(y).$$

(4) After Berry define for D, E be cpos:

.f: D→E is stable' if it is continuous and $\forall d\ e.\ e \le f(d) \Rightarrow \exists\ \min(\downarrow d \cap f^{-1}(\uparrow e))$.

(a) Observe that this definition makes sense on any cpo. (b) Show that in general the category of cpos and stable' maps is not even cartesian. Hint: consider the example of finite cpo that fails to be a $cpo_\wedge$ given in note (1). Call this domain D and define a pair of maps f: D→O and g: D→O (O Sierpinski space) such that the pairing fails to be stable'. (c) In general stable' implies stable. The converse holds on ω-algebraic $Cpo_\wedge$ with property I (introduced later in this chapter).

(5) Prove that $\mathbf{Cpo}_\wedge$ has limits of ω^{op}-diagrams.

(6) Develop a theory of stable, partial maps by analogy with the continuous case (warning: this is more a project than an exercise). Discuss lifting and sum in this framework.

(7) The partial operation of composition on meet-cpos is meet-preserving. Consider $\circ : (E\to F) \times (D\to E) \rightharpoonup (D\to F)$ the usual functional composition where D, E, and F are meet-cpos. Suppose (f, g), (f', g') ≤ (f'', g''). We have to show: $(f\circ g) \wedge (f'\circ g') = (f\wedge f')\circ(g\wedge g')$, that is for any d in D:

$f(gd)\wedge f'(g'd) = f(gd)\wedge f(g'd)\wedge f'(gd)\wedge f'(g'd)$.

It is enough to observe: $f(gd) = f(g''d)\wedge f''(gd)$ $\quad f(g'd) = f(g''d)\wedge f''(g'd)$

$f'(gd) = f'(g''d)\wedge f''(gd)$ $\quad f'(g'd) = f'(g''d)\wedge f''(g'd)$. □

Next we apply the "bifinite approach" to obtain a ccc of ω-algebraic cpos and stable maps denoted with $\mathbf{Bif}_\wedge$.

Proposition *(Stable Projections)*

Let D be a meet cpo and $p, q \lesssim id_D$. Then:

(1) $p \circ q = p\wedge q$.

(2) p is a projection.

(3) im(p) is downward closed.

Proof

(1) Remark first that since p and q are bounded their glb exists. Next observe:
$qd \le d \Rightarrow p(qd) = pd\wedge qd = (p\wedge q)(d)$.

(2) For p=q we obtain from (1): $p(pd) = pd\wedge pd = pd$.

(3) $d\le pd' \Rightarrow pd = p(pd')\wedge d = pd'\wedge d = d$. □

Definition *(Stable Bifinite)*

Let D be a meet cpo. D is a stable bifinite, and we will write $D\in \mathbf{Bif}_\wedge$, if there is a chain of (stable) projections $\{p_n\}_{n\in\omega}$ such that $im(p_n)$ is finite and $\sqcup_{n\in\omega} p_n = id_D$.

Proposition *($Bif_\wedge$ is a ccc)*

The category $\mathbf{Bif}_\wedge$ of stable bifinites and stable maps is cartesian closed.

Proof

Since meet cpos and stable maps form a ccc we only need to check that if D, $E\in \mathbf{Bif}_\wedge$ then $D\times E$, $D\to E\in \mathbf{Bif}_\wedge$. Let $\{p_n\}_{n\in\omega}$ and $\{q_n\}_{n\in\omega}$ be the sequences on D and E respectively. Let us verify that:

$\{\lambda(d, e).(p_n(d), q_n(e))\}_{n\in\omega}$ and $\{\lambda f.\lambda d.q_n(f(p_n(d)))\}_{n\in\omega}$

define the needed sequences of stable projections on $D\times E$ and $D\to E$ respectively. By the properties of stable projections p is a projection iff $p \lesssim id$. We compute:

• $(d, e) \le (d', e') \Rightarrow (p_n(d), q_n(e)) = (p_n(d')\wedge d, q_n(e')\wedge e) = (p_n(d'), q_n(e'))\wedge(d, e)$.

• After expansion we have to show:

$f \lesssim g \Rightarrow \alpha(d) \equiv q_n(f(p_n(d))) = q_n(g(p_n(d))) \wedge f(d) \equiv \beta(d)$. Observe:

$f(p_n(d))\le f(d), q_n \lesssim id \Rightarrow q_n(f(p_n(d))) = q_n(f(d))\wedge f(p_n(d))$ and

$f(d)\le g(d), q_n \lesssim id \Rightarrow q_n(f(d)) = q_n(g(d))\wedge f(d)$. $p_n(d)\le d, f \lesssim g \Rightarrow f(p_n(d)) = f(d)\wedge g(p_n(d))$.

Therefore: $\alpha(d) = q_n(g(d))\wedge f(d)\wedge g(p_n(d))$. On the other hand:

$g(p_n(d))\le g(d), q_n \lesssim id \Rightarrow \beta(d) = q_n(g(d))\wedge g(p_n(d))\wedge f(d)$. So $\alpha(d) = \beta(d)$.

For the finiteness of the images observe:

$im(\lambda(d, e).(p_n(d), q_n(e))) \cong p_n(D)\times q_n(E)$, $im(\lambda f.\lambda d.q_n(f(p_n(d)))) \cong p_n(D)\to q_n(E)$. □

2. A Characterization of Stable Bifinites

The main result here is a characterization of the objects in $\mathbf{Bif}_\wedge$ (cf. chapter 3 for an analogous result in the continuous case). Roughly they are ω-algebraic $\text{cpos}_\wedge$ satisfying a combination of properties M and I that we call (MI)*, where property M guarantees that each finite collection of compacts has a finite number of minimal upper bounds (mubs), and property I requires that the principal ideal generated by a compact element is finite. Then, in first approximation, the combined property (MI)* consists in asking that the iteration of the operator that computes the mubs and the operator that computes the principal ideals on a finite collection of compacts returns a finite collection.

Definition *(property I)*

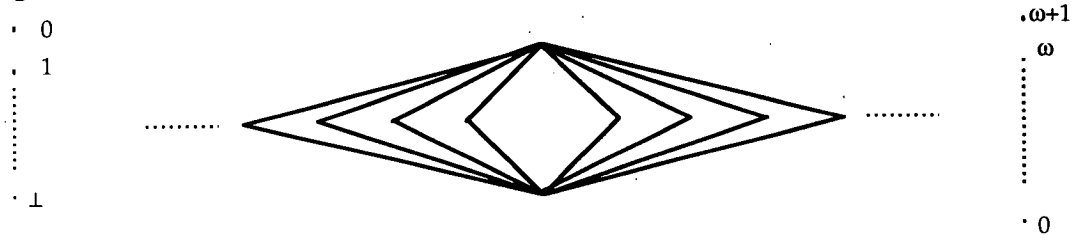
A cpo D has *property I* if for each compact, $d \in D_o$, the principal ideal $\downarrow d \triangleq \{e \mid e \leq d\}$ is finite. It is convenient to decompose property I in simpler properties.

Definition *(properties I_1, I_2, I_3)*

Let D be a cpo. We say that it has:

- *property* I_1 if every decreasing sequence of compacts is finite:
$$(\{x_n\}_{n\in\omega} \subseteq D_o \wedge \forall n \in \omega . x_n \geq x_{n+1}) \Rightarrow \{x_n\}_{n\in\omega} \text{ finite.}$$
- *property* I_2 if every increasing sequence of compacts under a compact is finite:
$$(x \in D_o \wedge \{x_n\}_{n\in\omega} \subseteq D_o \wedge \forall n \in \omega .\ x_n \leq x_{n+1} \leq x) \Rightarrow \{x_n\}_{n\in\omega} \text{ finite.}$$
- *property* I_3 if the immediate predecessors of a compact are finite:
$$x \in D_o \Rightarrow \text{Pred}(x) \triangleq \{y \in D \mid y < x \wedge y \leq y' < x \Rightarrow y = y'\} \text{ is finite.}$$

Example Three basic situations in which property I can fail:



Decreasing chain,$\neg I_1$ ω-branching,$\neg I_3$ Increasing chain,$\neg I_2$

Proposition *(decomposition property I)*

Let D be an algebraic cpo. Then it has property I iff it has properties I_1, I_2, and I_3.

Proof

(⇒) Observe that the sequences and the immediate predecessors are contained in ↓d, d being an appropriately chosen compact.

(⇐) Let $d \in D_o$. First observe that $\downarrow d \subseteq D_o$. If there is a non compact element, say x, under d then since D is an algebraic cpo $\downarrow x \cap D_o$ is directed and $\bigsqcup \downarrow x \cap D_o = x$. So we can build an ascending chain under d contradicting I_2.

Again from property I_2 follows that Pred(d) is complete in the sense that:

$e < d \Rightarrow \exists e' \in Pred(d).\ (e \leq e' < d)$. Otherwise we can again build a growing chain under d.

From property I_1 we conclude that by iterating the Pred operator every given element $e \in \downarrow d$ can be reached in a finite number of steps. If not using the completeness of Pred we build a decreasing chain included in $\uparrow e \cap \downarrow d$.

Now define: $X_0 \triangleq \{d\}$, $X_{n+1} \triangleq \cup\{Pred(x) \mid x \in X_n\} \cup X_n$. Then: (a) $\cup_{n\in\omega} X_n = \downarrow d$. (b) $\exists n \in \omega.\ X_{n+1} = X_n$, o.w. we contradict property I_1. (c) By property I_3, $\forall x \in D_o$. Pred(x) is finite. Hence: $\forall n \in \omega.\ X_n$ is finite. That implies: $\downarrow d$ is finite.[22] □

Property M

Let us recall some of the notions introduced in chpt. 3. Given (P, ≤) poset, and $X \subseteq P$ subset, we say that MUB(X) is *complete* if each upper bound of X is greater or equal than some element of MUB(X). We say that *X has property M* if MUB(X) is finite and complete. Next define the operator U as follows:

$$U(X) \triangleq \cup\{MUB(Y) \mid Y \subseteq_{fin} X\}$$

and denote with $U^*(X)$ the least set containing X and closed w.r.t. the U operator. If D is an algebraic cpo then we say that *D has property M* if

$\forall X \subseteq_{fin} D_o$. MUB(X) has property M.

The following fact is proved by induction on the cardinality of X (see also chpt. 3).

Fact: If $\forall x,y \in P$.MUB({x,y}) has property M then $\forall X \subseteq_{fin} P$.MUB(X) has property M.

Lemma *($Bif_\wedge \Rightarrow$ property I, property $I_1 \Rightarrow$ mub-completeness)*

(1) If D is a stable bifinite then D is an ω-algebraic $cpo_\wedge$ satisfying property I.

(2) If D is an algebraic cpo satisfying property I_1 then for each finite collection X of compacts in D, MUB(X) is complete.

Proof

(1) Let $\{p_n\}_{n\in\omega}$ be the sequence related to D. Observe that $d \in D$ is compact iff $\exists n.\ p_n(d) = d$. This gives ω-algebraicity. As for property I observe: $d' \leq d = p_n(d) \Rightarrow d' \in im(p_n)$ that is finite.

(2) Given any y, upper bound for X, there is a compact $y' \leq y$ that is also an upper bound for X. By the property I_1 there is $y'' \leq y'$ such that $y'' \in MUB(X)$. Otherwise we build an infinite decreasing chain under y'. □

Lemma *(mubs are pairwise incompatible)*

Let D be an algebraic $cpo_\wedge$. Then

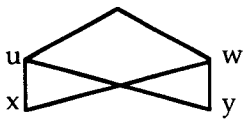
$$\forall X \subseteq_{fin} D_o.(\ \{y_1, y_2\} \subseteq MUB(X) \Rightarrow (y_1 = y_2 \vee \neg(y_1 \uparrow y_2))\ ).$$

Proof

If $y_1 \uparrow y_2$ then $\exists\ y_1 \wedge y_2 \in MUB(X)$. This forces $y_1 = y_2$. In other words the collection of mubs of a finite set is always composed of pairwise incompatible elements. □

[22] This is the contrapositive of König's lemma adapted to directed acyclic graphs.

Example: Here is a simple example suggesting that mubs have to be incompatible in $Cpo_{\wedge}$. Observe that $u\uparrow w$ but $\neg\exists(u\wedge w)$.



Lemma *(the U operator collapses at the second level)*

Let D be an algebraic meet cpo satisfying property I_1. Then

$\forall X \subseteq D_o$. $U(U(X)) = U(X)$.

Proof

By taking the mub of singleton subsets of U(X) one proves that $U(X) \subseteq U(U(X))$. Vice versa observe that $z \in U(U(X))$ iff $\exists\, Y \subseteq_{fin} U(X).(z \in MUB(Y))$. Say $Y \triangleq \{y_1,..., y_n\}$, $(n \geq 0)$. By definition of U(X) we have:

$y_i \in MUB(X_i)$ for $X_i \subseteq_{fin} X$ and i=1,..., n.

We want to show that $z \in MUB(X_1 \cup ... \cup X_n)$. Then by definition $z \in U(X)$. Certainly $z \in UB(X_1 \cup ... \cup X_n)$. Suppose $\exists z' \in UB(X_1 \cup ... \cup X_n)$. $(z' < z)$. Then $z' \notin UB(Y)$ as $z \in MUB(Y)$. Thus $\exists j \in \{1,.., n\}$. $\neg(y_j \leq z')$ then, by completeness of $MUB(X_j)$ (induced by property I_1), $\exists y_j' \in MUB(X_j)$. $(y_j' \leq z')$. So $y_j' \neq y_j$. But $y_j' \leq z' \leq z \geq y_j$, i.e $y_j' \uparrow y_j$. Then by the previous lemma on the incompatibility of mubs $y_j' = y_j$. Contradiction. □

Notes

(1) To summarize, if D is an algebraic $cpo_{\wedge}$ satisfying property I_1 then:

$\forall X \subseteq_{fin} D_o$. MUB(X) is complete and (U(X) finite $\Rightarrow$ U*(X) finite).

Roughly only one of the three situations considered in Smyth's theorem (chpt. 3) can arise, namely that MUB(X) is infinite.

(2) Suppose $D \in Bif_{\wedge}$. What additional information can we gather about the structure of compact elements? First, there are countably many since they are in the finite image of a countable chain of projections. Also let X be a finite collection of compacts, then we know that there is some stable projection p_n with finite image such that $X \subseteq im(p_n)$. The image of stable projections are downward closed, so the principal ideals generated by elements in $im(p_n)$ are contained in $im(p_n)$. Moreover, as it will be formally shown below, $im(p_n)$ is closed w.r.t. the U operator. Towards a characterization of $Bif_{\wedge}$ objects we are therefore led to the following

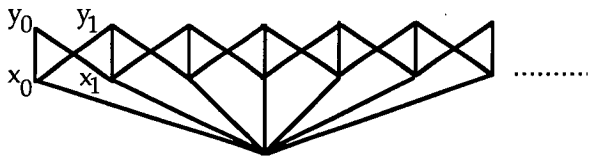
Definition *((MI)* property)*[23]

Given $(P, \leq)$ poset, and $X \subseteq P$ subset we recall that: $\downarrow(X) \triangleq \cup\{\downarrow x \mid x \in X\}$ and $U\downarrow(X) \equiv U(\downarrow(X))$. Let $(U\downarrow)^*(X)$ be the least set containing X and closed w.r.t. the $U\downarrow$ operator. Then we say that X has property (MI)* if $(U\downarrow)^*(X)$ is finite. If D is an algebraic $cpo_{\wedge}$ then we say that D has property (MI)* if for all $X \subseteq_{fin} D_o$, X has property (MI)*.

[23] The connection between stable projections and property (MI)* is also noted in Droste&Göbel[90].

Example *((MI)* strictly implies M and I)*

Let D be an algebraic $cpo_\wedge$. If D has property (MI)* then it also has property I and property M as if $x, y \in D_o$ then, respectively, $\downarrow x \subseteq (U\downarrow)^*(\{x\})$ and $U(\{x, y\}) \subseteq (U\downarrow)^*(\{x, y\})$. On the other hand the following domain Δ provides an example of an ω-algebraic $cpo_\wedge$ with properties I and M but without property (MI)*.



Observe that $(U\downarrow)^*(\{y_i\}) = \Delta$ and y_i is compact. Hence this domain does not belong to $Bif_\wedge$. Otherwise there should be a stable projection p_n with finite image and such that $p_n(y_i) = y_i$. But then by the previous note (2) we would have $(U\downarrow)^*(\{y_i\}) \subseteq im(p_n)$. Contradiction. As a matter of fact this example also shows that the iteration of the $U\downarrow$ operator does not need to collapse at any finite level.[24]

Theorem *(Characterizing Stable Bifinites)*

A cpo D is stable bifinite iff it is an ω-algebraic meet cpo with property (MI)*.

Proof

($\Rightarrow$) Let $\{p_n\}_{n\in\omega}$ be the chain of projections associated to D. We already know that D is an ω-algebraic $cpo_\wedge$ with property I. Let $X \subseteq_{fin} D_o$ and n such that $X \subseteq im(p_n)$ ($\equiv D_n$). We observe $MUB_D(X) = MUB_{Dn}(X) \subseteq D_n$. This follows from:

$$x \in UB_D(X) \Rightarrow p_n(x) \in UB_{Dn}(X) \cap UB_D(X).$$

That forces: $x \in MUB_D(X) \Rightarrow p_n(x) = x$. Now $U(X) = \cup\{MUB(Y) \mid Y \subseteq_{fin} X\}$. If Y is empty then $\sqcup Y = \perp$ and $p_n(\perp) = \perp$. Therefore we can infer:

$$X \subseteq im(p_n) \Rightarrow U(X) \subseteq im(p_n).$$

Next recall that $im(p_n)$ is downward closed, hence:

$$X \subseteq im(p_n) \Rightarrow \downarrow(X) \subseteq im(p_n).$$

Since $im(p_n)$ contains X and is closed w.r.t the $U\downarrow$ operator we conclude $(U\downarrow)^*(X) \subseteq im(p_n)$, that is finite.

($\Leftarrow$) Let $\{d_i\}_{i\in\omega}$ be an enumeration of D_o. Define: $X_n \triangleq \{d_0, ..., d_n\}$, and $D_n \triangleq (U\downarrow)^*(X_n)$, that is finite by assumption. Observe that D_n is downward closed, moreover given $d \in D$ $\downarrow d \cap D_n$ is directed by the definition of the U operator and the mub completeness. Therefore we can set:

$$p_n(d) \triangleq \max(\downarrow d \cap D_n) \equiv d_n$$

Let us check that $\{p_n\}_{n\in\omega}$ provides the needed chain:

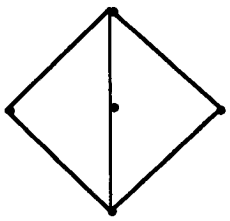
- p_n is continuous. Let X directed in D. Since D_n is finite $\exists x \in X.(\sqcup X)_n = x_n$.
- p_n is a projection with a finite image. Clearly $im(p_n) = D_n$ that is finite. We prove $p_n \lesssim id$. Let $x \leq y$. We check $x_n = y_n \wedge x$. Of course $x_n \leq y_n \wedge x$. If $z \leq y_n$ and $z \leq x$ then $z \leq x$ and $z \in D_n$ and this implies $z \leq x_n$. This also shows that p_n is stable being continuous and less than the identity. Hence p_n is a projection.

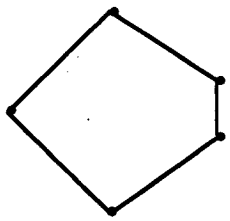
[24] It will be shown in A.1 that: (1) id_Δ is compact, (2) $\Delta \to \Delta$ has not property I_2, (3) $(\Delta \to \Delta) \to (\Delta \to \Delta)$ is not ω-algebraic.

• $p_n \lesssim p_{n+1}$. Let $x \leq y$. We check $x_n = y_n \wedge x_{n+1}$. Of course $x_n \leq y_n \wedge x_{n+1}$. If $z \leq y_n$ and $z \leq x_{n+1}$ then $z \leq x$ and $z \in D_n$ and this implies $z \leq x_n$.
• $\sqcup\{p_n\}_{n\in\omega} = id$. Observe: $\forall d \in D_o . \exists n . p_n(d) = d$. □

Notes

(1) Let D be a bounded complete algebraic cpo.[25] D is *distributive* if $\forall d, d'. d \uparrow d' \Rightarrow (d \vee d') \wedge e = (d \wedge e) \vee (d' \wedge e)$. A *dI-domain* is an algebraic cpo that is bounded complete, distributive, and has property I. The following are simple examples of non distributive finite lattices.





(2) Every countably based dI-domain is a stable bifinite. Hint: let D be a dI-domain and $\{d_i\}_{i\in\omega}$ an enumeration of D_o. Define the projections as

$$p_n(d) \triangleq \sqcup\{d_i \wedge d \mid 1 \leq i \leq n\}.$$

(3) *(Winskel)* Let D be a bounded complete ω-algebraic cpo. We say that $p \in D$ is *prime* if $\forall X \subseteq D. (\exists \sqcup X \wedge p \leq \sqcup X \Rightarrow \exists x \in X.(p \leq x))$. D is *prime algebraic* if $\forall x \in X. \exists \sqcup\{p \mid p \leq x, p \text{ prime}\} = x$. Show that a bounded complete cpo with property I is distributive iff it is prime algebraic (this is not easy).

(4) More examples of ccc of domains and stable maps can be found in the literature. E.g. Event Structures (Winskel), Qualitative Domains and Coherence Spaces (Girard).

(5) *(Zhang).* Let D be an ω-algebraic $cpo_\wedge$ satisfying property I. (a) Characterize $\{f^{-1}(\top) \mid f: D \to O, \text{ stable}\}$, O Sierpinski space. (b) Call these sets *stable neighborhoods*. Prove that they are closed by intersection but not by union. (c) Show that there is no topology for which the stable maps are continuous. Hint: consider the stable maps from O×O to O. There are four possible choices of a topology for O. Show that for each choice the stable and continuous maps do not coincide. (d) Characterize stable maps as those maps that preserve stable neighborhoods by inverse image.

(6) Property I is lost at higher order when considering the pointwise ordering. Example: take $(\omega^\perp \to O) \to O$, where O is Sierpinski space and $\omega^\perp$ are the natural numbers with the flat ordering. Define: $f_n(x) \triangleq$ if $x \leq n$ then $\top$ else $\perp$. Then $\{f_n\}_{n\in\omega}$ is a growing sequence of compacts in $(\omega^\perp \to O)$. Consider $step_{f_0,\top}$, this is a compact in $(\omega^\perp \to O) \to O$. We have: $step_{f_0,\top} > step_{f_1,\top} > \ldots > step_{f_n,\top} > \ldots$

(7) It is interesting to observe the behavior of the stable ordering. E. g.: (a) If D is

[25] We recall that a bounded complete algebraic cpo has glb of every non empty set.

a complete, ω-algebraic, lattice, with property I then: $D\to O \cong D_o^{\perp}$, where: O is Sierpinski space and $D_o^{\perp}$ is the collection of compact elements with the flat ordering. Hence all these countable lattices are collapsed by the operator: $\_\to O$.
(b) On the other hand if we take the exponent of a flat ordering we get:
$E^{\perp}\to O \cong (2^E, \subseteq) \oplus O$, where $\oplus$ is the coalesced sum.

3. Traces of Stable Functions

Let us start by observing that under the hypothesis that domains satisfy property I it is possible to give an alternative characterization of stable maps that follows Berry's original intuition. The trace of a stable map over algebraic meet cpos with property I plays a bit the role of the graph of a continuous function over algebraic cpos. Most of the proofs although lengthy are routine, this is why we omit some details.

Lemma *(stability and minimal points)*
Let D, E be meet cpos and f: D→E be continuous. Then:
(1) If $\forall d\in D.\ \forall e\in E.\ (e\le f(d) \Rightarrow \exists \min(\downarrow d \cap f^{-1}(\uparrow e))$ then f is stable.
(2) If D, E are algebraic $cpo_\wedge$ with property I and f: D→E is stable then:
$\forall d\in D.\ \forall e\in E.\ (e\le f(d) \Rightarrow \exists \min(\downarrow d \cap f^{-1}(\uparrow e))$.[26]

Proof
(1) Suppose $d\uparrow d'$. By monotonicity $f(d\wedge d') \le f(d)\wedge f(d')$. Vice versa let $d,d'\le d''$. Then $e \equiv f(d)\wedge f(d') \le f(d), f(d'), f(d'')$. So $\exists z.\ (z = \min(\downarrow d'' \cap f^{-1}(\uparrow e)))$. But then:
$z \le d, d' \Rightarrow z \le d\wedge d' \Rightarrow e \equiv f(d)\wedge f(d') \le f(z) \le f(d\wedge d')$.
(2) First we prove (2) for $(d,e)\in D_o\times E_o$. Suppose $e\le f(d)$. Since $\downarrow d$ is well founded there are minimal elements in $\downarrow d \cap f^{-1}(\uparrow e)$. Suppose m and m' are two such minimal. Then: $f(m\wedge m') = f(m)\wedge f(m') \ge e$. So $m = m\wedge m' = m'$.
Now prove (2) for $(d,e)\in D\times E_o$. Observe that also in this case local minima are compact. Finally consider the general case. □

Definition *(trace)*
Let D, E be algebraic meet cpos with property I and f: D→E be stable. Then we define the *trace* of f as: $tr(f) \triangleq \{(\min(\downarrow d \cap f^{-1}(\uparrow e)), e) \mid e \le f(d), (d, e)\in D_o \times E_o\}$.

[26] We sometime refer to $\min(\downarrow d \cap f^{-1}(\uparrow e))$ as local minimum.

Lemma *(some properties of traces)*

Let C, D, E be algebraic meet cpos with property I. Then:

(1) If f: D→E is a stable map then tr(f) is well-defined and tr(f) ⊆ graph(f), where graph(f) ≜ {(d, e)∈ D_o×E_o | e≤f(d)}.

(2) If f, g: D→E are stable maps then f≲g iff tr(f) ⊆ tr(g).

(3) If f, g: D→E are stable maps and f↑g then tr(f∧g) = tr(f)∩tr(g).

(4) If F is a directed set in D→E and g is stable then g = ⊔F iff tr(g) = ∪{tr(f) | f∈F}.

(5) If f: D→E is a stable map and tr(f) is finite then f is a compact in the functional space. In particular every step function is stable and compact.

(6) If f: C→D, g: D→E are stable and (c, d)∈ tr(f), (d, e)∈ tr(g) then (c, e)∈ tr(g∘f).

Proof

(1-5) These proofs are left as exercise.

(6) By assumption f(c) ≥ d and g(d) ≥ e. By composing we have g(fc) ≥ e. Suppose c'≤ c and g(fc') ≥ e. Then we have fc', d ≤ fc and e ≤ g(fc'), gd. Since d is a local minimum d ≤ fc'. Moreover c is also a local minimum so c' = c. □

Can we characterize compact stable maps as those having finite trace? The answer is yes for stable bifinites (exercise!), and in a "locally distributive case" presented in A.2. The answer is no in general, as shown by the following

Example: Consider the domain Δ defined in section 2. Clearly tr($id_Δ$) is infinite. We want to show that $id_Δ$ (≡$_{abr.}$id) is a compact element of the functional space.

Say that $f_{|k} = id_{|k}$ if ∀i≤k. f(x_i) = x_i, f(y_i) = y_i. Observe:

$f_{|k} = id_{|k}$, $g_{|k+1} = id_{|k+1}$, f≲g ⇒ $f_{|k+1} = id_{|k+1}$

Because: f≲g , x_{k+1}≤ y_k ⇒ f(x_{k+1}) = y_k∧x_{k+1} = x_{k+1}

f≲g , x_k≤ y_{k+1} ⇒ x_k = f(x_k) = f(y_{k+1})∧ x_k ⇒ x_k≤f(y_{k+1})

x_{k+1} = f(x_{k+1}) ≤ f(y_{k+1}) ≤ y_{k+1} ⇒ f(y_{k+1}) = y_{k+1}.

One concludes that id will appear in every chain that converges to id (that is a maximal element). So id is compact.

As a final remark we define an infinite growing chain below id. Let $\{z_i\}_{i∈ω} ⊂ \{x_i\}_{i∈ω}$ and such that ∀i, j∈ ω.(i≠j ⇒ ¬(z_i↑z_j)). Define

$p_k(x)$ ≜ if ∃i≤k. (z_i = x_j ∧ (x = x_j ∨ x = y_j)) then z_i else ⊥

Observe: (a) p_k is well-defined. (b) p_k is monotone. (c) p_k≲id. (d) p_k≲p_{k+1}. Therefore Δ→Δ has not property I_2. From a lemma stated in A.1 follows that (Δ→Δ)→(Δ→Δ) is not ω-algebraic. □

4. Some Domain Theoretic Constructions

In this section we want to support the view that categories of domains and stable maps provide *a theory of finite approximations* that can stand a comparison with the standard theory based on the notion of continuous map.

$\mathbf{Bif}_{\wedge}^{ep}$ as an ω-Algebroidal Category

In this section we verify that $\mathbf{Bif}_{\wedge}^{ep}$ is an *ω-algebroidal category* with the *amalgamation property*. Therefore it has a *universal homogeneous object*. The proof follows the same schema as for $\mathbf{Bif}^{ep}$ in chpt. 5.

Theorem(*$Bif_{\wedge}^{ep}$ has a Universal Homogeneous Object)*

Let $\mathbf{Bif}_{\wedge}^{ep}$ be the category of stable bifinite domains and stable embedding projection pairs. Then

(1) $\mathbf{Bif}_{\wedge}^{ep}$ is an ω-algebroidal category and the collection of compact objects has the amalgamation property.

(2) $\mathbf{Bif}_{\wedge}^{ep}$ has a universal homogeneous object.

Proof

(1) Let us just show that $\mathbf{Bif}_{\wedge}^{ep}$ has the amalgamation property. Consider three finite posets $(E, \leq)$, $(D_1, \leq_1)$, $(D_2, \leq_2)$ with maps $h_i: E \to D_i$, $i \in \{1,2\}$, in $\mathbf{Bif}_{\wedge}^{ep}$. W.l.o.g. assume $E = D_1 \cap D_2$, then:

$$\forall e, e' \in E.\ (e \leq e' \Leftrightarrow e \leq_1 e' \Leftrightarrow e \leq_2 e').$$

Now we define the amalgam as the set $F \triangleq E \cup (D_1 \setminus E) \cup (D_2 \setminus E)$. It is helpful to recall that E is downward closed in D_i so we define:

$f \leq_F f' \Leftrightarrow f, f' \in D_i,\ f \leq_i f'$ where i=1 or i=2.

Verify that $\leq_F$ is a partial order. We are left with the definition of the morphisms $k_i: D_i \to F$, $i \in \{1, 2\}$. Take the inclusions for k_i^+. Define:

$k_1^-(f) \triangleq$ if $f \in D_1$ then f else $h_2^-(f)$. k_2^- is defined symmetrically. Verify: (a) k_i is a morphism in $\mathrm{Bif}_{\wedge}^{ep}$. (b) $k_1^- \circ h_1 = k_2^- \circ h_2$. One has basically to check that F has glbs of compatible elements and that k_i^- is a stable projection.

(2) $\mathrm{Bif}_{\wedge}^{ep}$ is an ω-algebroidal category with amalgamation property therefore we can apply the theorem in chpt. 5. □

A Quick Construction of a Retraction of all Retractions for Stable Bifinites

Scott (Scott[80]) shows that the collection of finitary retractions over a bounded complete algebraic cpo D is the image of a finitary retraction over the space of continuous functions $D \to D$. Rothe (Rothe[91]) extends this result to a class of FS-domains (see Jung[91]). In the stable case Berardi (Berardi[91]) was apparently the first to observe that when working over dI-domains the image of a stable retraction is still a dI-domain. It was then possible to adapt Scott's technique to show that the space of retractions over a dI-domain is a retract of the functional space In the sequel we give the corresponding of Berardi's results for 'stable bifinites'. The proof is new and shorter, this seems due to the fact that stable bifinites can be described in a synthetic way as 'directed colimits of stable projections'.

We refer to Amadio&Longo[87] and Berardi[91] for the origins of the problem. We will simply mention here that when the domain at hand is a model of the type free lambda calculus it is easy to build models for type theories with a type of all types, by interpreting types-as-retractions (see exercise tp:tp in chpt. 6).

Proposition *($D\in Bif_\wedge$, r retraction $\Rightarrow$ $r(D)\in Bif_\wedge$)*

Let D be a stable bifinite and r: D→D be a stable retraction then the image, r(D), is a stable bifinite.

Proof

Let $\{p_i\}_{i\in I}$ be a directed collection of projections associated to D. Define $q_i = r\circ p_i\circ r$, for $i\in I$. Observe $q_i \preceq r\circ id\circ r = r$, and $im(q_i)$ is finite. Since $\bigsqcup_{i\in I} q_i = r$, we conclude r(D) is a stable bifinite. □

We give a simple proof of the fact that the collection of stable retractions over a stable bifinite D, say Ret(D), is a retract of its stable functional space D→D. The *keyvault of the construction* is to observe that given f: D→D, with im(f) finite, there is a natural way to associate to f a retraction, namely iterate f a finite number of times (this point is also used in Rothe[91]).

Theorem *($D\in Bif_\wedge \Rightarrow Ret(D) \lhd D\to D$)*

Given a stable bifinite D the collection of stable retractions, Ret(D), is a retract of the functional space D→D.

Lemma

Let D be a stable bifinite with the relative directed set of projections $\{p_i\}_{i\in I}$. Then for any f: D→D, $\#\{(f\circ p_i\circ f)^k \mid k\geq 1\}\cap Ret(D) = 1$.

Proof

First let us recall a simple fact about combinatorics.

Let X be a set, g: X→X a function, and im(g) finite then $\#\{g^k \mid k\geq 1\}\cap Ret(X) = 1$.

Because: $\forall k\geq 1.\ im(g^{k+1}) \subseteq im(g^k)$. Since im(g) is finite the following is well defined: $h \triangleq \min\{k\geq 1 \mid im(g^{k+1}) = im(g^k)\}$. So $g_{\restriction im(g^h)}$ is a permutation. If $\#im(g^h) = n$ then $(g^h)^{n!}$ is the identity on $im(g^h)$, and therefore a retraction over X. As for the uniqueness observe that if $g^i\circ g^i = g^i$ and $g^j\circ g^j = g^j$ for i, j≥1 then $g^i = g^{ij} = g^j$.

Next observe that $im(p_i)$ finite implies $im(f\circ p_i\circ f)$ is finite, and we can apply the previous fact. □

Conventions: In the hypotheses of the previous lemma we write:

$$f_i \equiv f\circ p_i\circ f, \qquad k_i \equiv \#im(p_i)\ !$$

Note that k_i is an upper bound on the least k such that $f_i^k\in Ret(D)$, and it is independent from f.

Next the crucial remark is that: $r\in Ret(D) \Rightarrow r_i\in Ret(D)$, because by the definition of stable order: $r_i\preceq r$, $r_i d\leq rd \Rightarrow r_i(r_i d) = r_i(rd) \wedge r(r_i d) = r_i d \wedge r_i d = r_i d$. It is then appealing trying to define:

$$\rho: (D\to D)\to Ret(D), \qquad \rho(f) = \bigsqcup_{i\in I} (f_i)^{k_i},$$

as: (i) the join of a directed set of retractions is a retractions, and (ii) $r \in Ret(D) \Rightarrow \rho(r) = \sqcup_{i\in I} (r_i)^{k_i} = \sqcup_{i\in I} r_i = r$. The following lemma concludes our argument by verifying that ρ is a stable retraction over $D\to D$.

Lemma

(1) Function composition is a stable operation.

(2) If D is a meet cpo then Ret(D) is a meet cpo (with the order given by $D\to D$).

(3) ρ is a stable map.

(4) $im(\rho) = Ret(D)$, and $\rho\circ\rho = \rho$.

Proof

(1) Let $f, g \lesssim h$, and $f', g' \lesssim h'$, so that $dom(f) = dom(g) = cod(f') = cod(g')$. We have to show: $(f\wedge g)(f'\wedge g')(d) = ((f\circ f')\wedge(g\circ g'))(d)$, any d. We compute:

$$(f\wedge g)(f'\wedge g')(d) = (f\wedge g)(f'd\wedge g'd) = f(f'd) \wedge f(g'd) \wedge g(f'd) \wedge g(g'd).$$

Observe: $f \lesssim h, g'd \le h'd \Rightarrow f(g'd) = f(h'd) \wedge h(g'd) \ge f(f'd) \wedge g(g'd)$.
$g \lesssim h, f'd \le h'd \Rightarrow g(f'd) = g(h'd) \wedge h(f'd) \ge g(g'd) \wedge f(f'd)$.

Hence: $(f\wedge g)(f'\wedge g')(d) = f(f'd) \wedge g(g'd) = ((f\circ f')\wedge(g\circ g'))(d)$, any d.

It is worth pointing out that for $f = f', g = g'$ we get: $(f\wedge g)^k = f^k\wedge g^k$, for $k\ge 1$.

(2) Note that $Ret(D) = Fixpoints(\lambda f.\ f\circ f)$. In complete analogy with the continuous case it is easy to show that the collection of fix-points of a stable map over a meet cpo is a meet cpo.

(3) First observe that for any given $f: D\to D$, $\{ (f_i)^{k_i} \}_{i\in I}$ is a *directed set*. Because:

$$p_i \lesssim p_j \Rightarrow (f_i)^{k_i} = (f_i)^{k_i k_j} \lesssim (f_j)^{k_i k_j} = (f_j)^{k_j}.$$

Therefore by (2) it follows that $\rho(f)$ is well-defined and it is a retraction. A similar argument shows that ρ is *monotone*, as: $f \lesssim g \Rightarrow (f_i)^{k_i} \lesssim (g_i)^{k_i}$, for any i.

Next we wish to show that ρ is *meet-preserving*. Suppose $f, g \lesssim h$. Observe:

$(f\wedge g)_i = (f\wedge g)\circ p_i\circ(f\wedge g) = (f\circ p_i\circ f) \wedge (f\circ p_i\circ g) \wedge (g\circ p_i\circ f) \wedge (g\circ p_i\circ g)$. Also, any d:

$f \lesssim h, p_i(gd) \le p_i(hd) \Rightarrow f(p_i(gd)) = f(p_i(hd)) \wedge h(p_i(gd)) \ge f(p_i(fd)) \wedge g(p_i(gd))$.

$g \lesssim h, p_i(fd) \le p_i(hd) \Rightarrow g(p_i(fd)) = g(p_i(hd)) \wedge h(p_i(fd)) \ge g(p_i(gd)) \wedge f(p_i(fd))$.

Hence: $(f\wedge g)_i = f_i\wedge g_i$. If we combine this with the stability of iteration (1), and property (3) of meet cpos we get:

$$\rho(f\wedge g) = \sqcup_{i\in I} ((f\wedge g)_i)^{k_i} = \sqcup_{i\in I} (f_i\wedge g_i)^{k_i} = \sqcup_{i\in I} ((f_i)^{k_i}\wedge(g_i)^{k_i}) =$$
$$\sqcup_{i\in I} (f_i)^{k_i} \wedge \sqcup_{i\in I} (g_i)^{k_i} = \rho(f) \wedge \rho(g).$$

It remains to show that ρ *preserves directed sets*. Let $\{ f_v \}_{v\in V}$ be a directed set in $D\to D$ (do not confuse f_v and f_i !). First observe:

$(\sqcup_{v\in V} f_v)_i = (\sqcup_{v\in V} f_v)\circ p_i\circ(\sqcup_{v\in V} f_v) = \sqcup_{v\in V} (f_v\circ p_i\circ f_v) = \sqcup_{v\in V} (f_v)_i$. Also:

$((\sqcup_{v\in V} f_v)_i)^{k_i} = (\sqcup_{v\in V} (f_v)_i)^{k_i} = \sqcup_{v\in V} ((f_v)_i)^{k_i}$. Hence:

$\rho(\sqcup_{v\in V} f_v) = \sqcup_{i\in I} ((\sqcup_{v\in V} f_v)_i)^{k_i} = \sqcup_{i\in I} \sqcup_{v\in V} ((f_v)_i)^{k_i} =$
$\sqcup_{v\in V} \sqcup_{i\in I} ((f_v)_i)^{k_i} = \sqcup_{v\in V} \rho(f_v)$.

(4) We have already argued that $r\in Ret(D) \Rightarrow \rho(r) = r$. This immediately implies the second claim, as: $\rho(f)\in Ret(D) \Rightarrow \rho(\rho(f)) = \rho(f)$. □

Stable Projections

By analogy with the continuous case (chpt. 5) we develop some properties of stable projections.

Proposition *($D\in Bif_\wedge \Rightarrow Prj(D)\in Bif_\wedge$)*

(1) Let D be a meet cpo and suppose p is a projection, $p \lesssim id$. If D is an (ω-)algebraic meet cpo (stable bifinite) then im(p) is an (ω-)algebraic meet cpo (stable bifinite).

(2) If D is a stable bifinite then $Prj(D) = \downarrow(id_D)$ is a stable bifinite and a lattice.

Proof

(1) Since im(p) is downward closed and $p(D)_o = p(D)\cap D_o$ we can infer that im(p) is an (ω-)algebraic $cpo_\wedge$. If $D\in Bif_\wedge$ and $\{q_n\}_{n\in\omega}$ is the corresponding chain of projections then we define $\{p\wedge q_n\}_{n\in\omega}$ as the chain related to im(p). Of course that D is bifinite also follows from the results on retractions.

(2) Immediate from the characterization of the compacts of a principal ideal and the fact that $Prj(D) = \downarrow(id_D)$. □

Notes

(1) The only stable closure is the identity! Actually the identity is always a maximal element in the stable order: $id \lesssim f \Rightarrow \forall x.\ (x \le fx) \Rightarrow \forall x.\ x = fx\wedge fx = fx$.

(2) Each projection (as well as each closure) uniquely determines a poset via its image. Let p, q be projections over a poset D. Then p(D) = q(D) implies p = q. As:

$qx \le x \Rightarrow qx = p(qx) \le px, \quad px \le x \Rightarrow px = q(px) \le qx.$

(3) Let D be the example of finite $cpo_\wedge$ not bounded complete given in note (1), section 1. Then it is not the case that $Prj(D) \sqsubseteq D\to D$.

Exercise REPR: By analogy with what was done in chpt. 5 study the representation problem of the functors over $\mathbf{Bif}_\wedge^{ep}$ as stable maps over Prj(U) (where U is some universal (homogeneous) domain). Show that product and exponent are representable.

Appendix 1. Functional Spaces and ω-algebraicity

In this appendix we state that property M and property I are necessary to enforce the *ω-algebraicity* of functional spaces. Proofs are omitted as they are too technical but can be found in Amadio[91]. More precisely it turns out that properties M, I_1, and I_2 are necessary. One can also show that property I_3 is necessary under a rather mild hypothesis. The necessity of property (MI)* is still open.

In the first place we want to guarantee that in any full subcategory of algebraic meet cpos if the terminal object, the product, and the exponent exist then they coincide up to isomorphism with the ones defined in $\mathbf{Cpo}_\wedge$. The proof is basically the same as in the continuous case (see chpt. 3).

Fact *(terminal object, product, and exponent are pre-determined)*

Let **C** be a full subcategory of algebraic meet cpos and stable maps. Then:

(1) If **C** has terminal object T then T is a one point cpo.

(2) If **C** has terminal object T and products $D \xleftarrow{\pi^D} D\otimes E \xrightarrow{\pi^E} E$ then $D\otimes E \cong D\times E$, the standard product in $\mathbf{Cpo}_\wedge$.

(3) If **C** has terminal object T, finite products $D \xleftarrow{\pi^D} D\otimes E \xrightarrow{\pi^E} E$, and exponent $E^D\otimes D \xrightarrow{ev_C} E$ then $E^D \cong D\Rightarrow E$, the standard exponent in $\mathbf{Cpo}_\wedge$.

Next we list a series of technical lemmas.

Lemma *(D, D→D ω-algebraic ⇒ D has property M)*[27]

If D and D→D are ω-algebraic meet cpos then $\forall x, x'\in D_o$. MUB({x, x'}) is finite.

Lemma *(D, D→D ω-algebraic ⇒ D has properties I_1, I_2)*

Suppose D and D→D are ω-algebraic meet cpos then D has properties I_1 and I_2.

Definition *(distributivity)*

Let D be a bounded complete algebraic cpo.[28] D is *distributive* if

$$\forall d, d'.\ d\uparrow d' \Rightarrow (d\vee d')\wedge e = (d\wedge e)\vee(d'\wedge e).$$

Lemma *(D,D→D ω-alg. + distributivity principal ideals ⇒ D has property I_3)*

Suppose D and D→D are ω-algebraic meet cpos. Then

(1) If $d\in D$ then $\downarrow d$ is an ω-algebraic lattice.

(2) If $d\in D_o$ and $\downarrow d$ is distributive then $\downarrow d$ is finite.

(3) If for each $d\in D_o$ $\downarrow d$ is distributive then D has property I.

---

[27] It will be shown in the next appendix that property M is not necessary if we do not ask for the countability of compact elements.

[28] See also note (1), §3. Note that a bounded complete algebraic cpo has glb of every non empty set.

Lemma *(D, D→D, and ↓d→↓d ω-algebraic ⇒ D has property I_3)*[29]

Suppose D and D→D are ω-algebraic meet cpos and for each $d \in D_o$ ↓d→↓d is an ω-algebraic meet cpo. Then D has property I_3 .

Appendix 2: L-domains in the stable case

In this appendix we introduce the main ideas about the construction of a stable version of L domains. The categories $L_\wedge$ and $L_{o\wedge}$ that we define are mentioned in Coquand[89] as the "poset case" of a more general categorical construction.

Conventions: A cpo D is an *Lcpo* if every principal ideal is a complete lattice. A cpo D is an *L-domain* if every principal ideal is an algebraic complete lattice. An *L_0-domain* is an ω-algebraic L-domain satisfying property M. Let us recall (chpt. 3):

Fact: The categories of Lcpos, L-domains, L_0-domains and continuous maps are cartesian closed.

We recall that in Lcpos we have *global glbs* of bounded, non empty sets. We also have *local lubs* of bounded sets, Henceforth when we compute the lub of a set X contained in ↓x we write $\sqcup^x X$. It might also br useful to recall that if D is an algebraic cpo then every principal ideal is algebraic and every local compact is also a global compact.

Three basic observations lead to prove the previous fact:

(1) If D, E∈ Lcpo, F⊆D→E, and F is bounded by f then we can define:

$g(d) \triangleq \sqcup\{h(d) \mid h \in F\}$

this is the *lub* and it is *continuous*. Hence Lcpo is a ccc.

(2) The problem with the algebraicity of the functional space is that the collection of compacts below an element may fail to be directed. In general we have

$f = \text{lub}_{D\to E}\{\text{step}_{d,e} \mid e \leq f(d), (d, e) \in D_o \times E_o\}$

and $\text{step}_{d,e}$ is always a compact in the functional space. If ↓f is a lattice then we can easily manufacture a directed collection of compacts in ↓f whose lub is f. Hence L-domain is a ccc.

(3) It is not difficult to prove that L-domains satisfy mub-completeness and the U-operator collapses at the second level. So ω-algebraic L-domains with property M are bifinites. Then L_0-domains form a ccc.

In the *stable case* we proceed by analogy. The first problem arise in proving that the function g in (1) is stable. There we seem to need *distributivity* (the set {h(d) | h∈ F} does not need to be directed!). Therefore we have

[29]So, roughly speaking, every full subccc of ω-algebraic $cpo_\wedge$ whose objects are closed by principal ideals has property I_3.

Definition *($Lcpo_\wedge$)*

D is a stable Lcpo *($Lcpo_\wedge$)* if it is a cpo and every principal ideal is a distributive lattice.[30] We denote with $\mathbf{Lcpo}_\wedge$ the category of $Lcpos_\wedge$ and stable maps.[31]

Proposition *($Lcpo_\wedge$ is a ccc)*

The category $\mathbf{Lcpo}_\wedge$ is cartesian closed.

Proof

Our concern is to show that principal ideals in the exponent are distributive lattices. Given D, E$\in$Lcpo$_\wedge$, F$\subseteq$D$\to$E, and F bounded by f we define:

$$g(d) \triangleq \sqcup^{f(d)}\{h(d) \mid h\in F\}$$

Verify g is continuous. We check g is stable. Suppose d, d'$\leq$d". Then

$$g(d\wedge d') = \sqcup^{f(d")}\{h(d\wedge d') \mid h\in F\} = \sqcup^{f(d")}\{h(d)\wedge h(d') \mid h\in F\} =$$
$$\sqcup^{f(d")}\{h(d) \mid h\in F\}\wedge\sqcup^{f(d")}\{h(d') \mid h\in F\} = g(d)\wedge g(d').$$

Verify g = $\sqcup^f$F. We also check distributivity of $\downarrow$f. Assume k$\leq$f then

$\sqcup^f F\wedge k(d) = \sqcup^{f(d)}\{h(d) \mid h\in F\}\wedge k(d) = \sqcup^{f(d)}\{h(d)\wedge k(d) \mid h\in F\} = \sqcup^{f(d)}\{h\wedge k(d) \mid h\in F\}$. □

We now wish to develop a category of algebraic meet cpos. For technical reasons we will use the trace characterization of the stable ordering. Then property I is also needed.

Definition *($L_\wedge$-domains)*

D is a stable L-domain *($L_\wedge$-domain)* if it is a cpo and every principal ideal is a dI-domain.[32] An *$L_{o\wedge}$-domain* is an $L_\wedge$-domain that is ω-algebraic and has property M.

We denote with $\mathbf{L}_\wedge$ ($\mathbf{L}_{o\wedge}$) the category of $L_\wedge$-domains ($L_{o\wedge}$-domains) and stable maps.

Lemma

Let D, E be $L_\wedge$-domains and f: D$\to$E be a stable map.

(1) If T$\subseteq$tr(f) then there is a stable map g such that g $\lessdot$ f and

$$tr(g) = \{(d', e') \mid (d', e') \leq (d, e), (d', e')\in tr(f), (d, e)\in T\}$$

Moreover if T is finite then tr(g) is finite and g is compact.

(2) If f is compact then tr(f) is finite.

Proof

(1) Define $T' \triangleq \{(d', e') \mid (d', e') \leq (d, e), (d', e')\in tr(f), (d, e)\in T\}$, one has just to verify that $g(x) \triangleq \sqcup\{d' \mid (d', e')\in T', d'\leq x\}$ is stable (this uses distributivity). Then it follows by definition that tr(g)$\subseteq$tr(f) and so g$\lessdot$f. By property I and the definition of tr(g) one also concludes that tr(g) is finite if T is finite. By the properties of traces follows that g is compact.

(2) Suppose tr(f) is infinite. Then, by (1), we can compute f as the lub of the

[30] A lattice is distributive if it is complete and it satisfies: $\cup X\wedge y = \cup\{x\wedge y \mid x\in X\}$.

[31] Verify that $Lcpo_\wedge$ is a subcategory of $Cpo_\wedge$.

[32] From this it follows that every principal ideal is an algebraic distributive lattice with property I.

"extensions" of the finite parts of its trace. Remark that this set is directed, hence f is not compact. □

We are now ready to simulate step (2) of the continuous case and state

Proposition *($L_\wedge$ is a ccc)*
The category $\mathbf{L}_\wedge$ is cartesian closed.

Proof
One has to verify that the functional space is algebraic and it has property I. Algebraicity follows from (1) of the previous lemma as we can always approximate a map as the extensions of the finite parts of its trace. Property I follows from (2) of the previous lemma. □

Note: Observe that cartesian closure of $L_{o\wedge}$ follows by the fact that $L_{o\wedge} \subset Bif_\wedge$ and exponentiation in $Bif_\wedge$ preserves property M and ω-algebraicity.

Note: P. Taylor has shown that also the larger category of L-domains and stable maps forms a ccc (the proof of this fact is rather complex). This is the largest ccc of algebraic meet cpos and stable maps.

8. Domains and Realizability

Contents: 1. A Universe of Realizable Sets, 2. Interpretation of System F, 3. Towards Recursion, 4. Separated Partial Equivalence Relations, 5. N-completeness.

In the last years a line of research has been developed that goes under the name of "synthetic domain theory" (see Hyland[91]). One of the main goals in this area is to build theories in which *data types can be regarded as sets*. A number of examples show that classical set theory is not well-suited to this purpose, e.g. think of models for recursive functions definitions, type-free lambda calculus, and polymorphism. However it has become more and more clear that this program can be achieved if one takes a "constructive attitude". In particular "realizability" has been the part of constructive mathematics that has been more successful in implementing this plan.

In this chapter we introduce some basic notions and results about realizability models at an elementary level. In particular the category of *partial equivalence relations* (pers) is introduced. This category is equivalent to a full subcategory of the *effective topos*, a universe of constructive sets built out of realizability interpretations. In this sense the category of pers fits our goal of regarding data types as (constructive) sets. The category of pers has very strong closure properties, in particular we will show how it is possible to model system F in it (cf. chpt. 6).

A satisfying theory of computation has to account for recursive definitions of functions and data. The category of pers is not adapted to this goal, however we will show that it is possible to define various *reflective* sub-categories of pers whose objects can be regarded as *intrinsically* partially ordered complete sets. Moreover their definition guarantees that all maps preserve the intrinsic order-theoretic structure. This last result can be regarded as a generalized form of the Myhill-Shepherdson theorem presented in chpt. 1.

Kleene Interpretation of HA

Kleene, in Kleene[45], first introduced a realizability interpretation of Heyting arithmetic (HA). This interpretation provides a standard link between *constructive mathematics* (as formalized in HA) and *classical recursion theory*. Moreover it has the merit of giving a solid mathematical content to the Brouwer-Heyting-Kolmogorov explanation of constructive proofs.

Consider Peano Arithmetic formalized in an Intuitionistic First Order Logic with Equality and a signature with symbols "0" for zero, "s" for successor, etcetera.

Let $\mathbb{N}$ be the intended interpretation of the signature over the structure of natural numbers. Define a *binary realizability relation between numbers and formulae*, $\Vdash \subseteq \omega \times \text{Form}$, by induction on the formulae, as follows:

$n \Vdash t=s$	if	$\mathbb{N} \vDash t=s$	(1)
$n \Vdash \bot$	if	never	
$n \Vdash A \wedge B$	if	$\pi_1 n \Vdash A$ and $\pi_2 n \Vdash B$	(2)
$n \Vdash A \vee B$	if	$(\pi_1 n=0$ and $\pi_2 n \Vdash A)$ or $(\pi_1 n=1$ and $\pi_2 n \Vdash B)$	
$n \Vdash A \rightarrow B$	if	for each m $(m \Vdash A$ implies $\{n\}m \Vdash B)$	(3)
$n \Vdash \forall x.A$	if	for each m $(\{n\}m \Vdash [\underline{m}/x]A)$	(4)
$n \Vdash \exists x.A$	if	$\pi_2 n \Vdash [\underline{\pi_1 n}/x]A$	

where: (1) $\vDash$ is the standard validity predicate for atomic formulae of arithmetic interpreted in the structure $\mathbb{N}$ of natural numbers. (2) π_1, π_2, are the first and second projections w.r.t. an injective coding $< , >: \omega \times \omega \rightarrow \omega$. (3) $\{n\}m$ is the n-th Turing machine applied to the input m. (4) $\underline{m}$ is a numeral in the system HA corresponding to the natural number m.

Applications

To any formula A in HA we can associate the set of its realizers:

$[\![A]\!] \triangleq \{n \mid n \Vdash A\}$.

It is easy to prove a soundness theorem saying that any provable formula has a non empty collection of realizers (i.e. it is realizable), and a consistency theorem saying that there are formulas with an empty collection of realizers (e.g. $[\![A \wedge \neg A]\!] = \varnothing$). However the most important applications of this interpretation concern the consistency proof of certain *extensions* of Heyting arithmetic. For instance Church Thesis, and Markov Principle, which are formalized in HA as follows:

(CT) $\forall n.\exists! m.\ A(n, m) \rightarrow \exists k.\forall n.\exists m.\ (A(n, Um) \wedge Tknm)$

where: Um is the final result of a computation m, and Tknm holds iff the program k with input n produces a terminating computation m.

(MP) $(\forall n.\ (\ A(n) \vee \neg A(n)\ ) \wedge \neg\neg \exists n.A(n)\ ) \rightarrow \exists n.\ A(n)$,

for A primitive recursive predicate, i.e. no unbounded quantifications.

A Π^0_2 sentence is a formula of the shape $\forall n.\exists m.\ A(n,m)$, where A is atomic. Informally, (CT) states that any single-valued relation over the natural numbers that is definable in HA is computable by some recursive function. The idea is that from any (constructive) proof of a Π^0_2 sentence, $\forall n.\exists! m.\ A(n,m)$, one can (provably) extract an algorithm that given n finds the m such that A(n, m).

On the other hand (MP) holds because if one has a decidable predicate $(\forall n.(A(n) \vee \neg A(n))$ and an oracle telling that this predicate is non-empty $(\neg\neg \exists n.A(n))$ then one can effectively find an element satisfying the predicate simply by enumerating the candidates and checking the predicate on them.

Model Theoretic Abstraction

We are now confronted to two main problems:

- We want to model Type Theories (not just HA).
- We want to interpret Proofs/Programs and not just Propositions/Types.

In order to obtain some results in this direction we will concentrate on a special class of *realizability models*. Two basic features of these models are:

- Types can be regarded as Constructive Sets.
- There is a distinction between a "typed value" and its "type-free realizers".

Let us elaborate a bit on these two points. A model for a programming language should include some notion of effectivity and/or computability. In the kind of models we are going to present the underlying realizability structure plays the role of an abstract machine. A datum is represented as a collection of extensionally equivalent implementations. A map from data A to data B is a function that can be actually implemented by a realizer and that transforms equivalent implementations of a datum in A into equivalent implementations of a datum in B (this aspect of the model is relevant for the interpretation of subtyping in programming languages, see Bruce&Longo[88], Amadio&Cardelli[90]).

The fact that all operations have to be realized leads to very interesting properties of the model reflecting its computational nature. A typical example is the validity of a "Uniformity Principle" which plays an important role in the interpretation of second order quantification as intersection.

There have been several attempts in this direction. It seems fair to say that this program was first pushed by Dana Scott and his students: McCarty [84], and Rosolini[86], whose work relates in particular to the *effective topos* (Hyland[82], but see also Mulry[81] for another approach). Recently there has been another wave of results in Amadio[89], Abadi&Plotkin[90], Freyd&al.[90], Phoa[90].

1. A Universe of Realizable Sets

In this section we start to put into practice our program of giving a generalization of Kleene realizability.

Realizability Structure

In Kleene interpretation the basic "realizability structure" is given by the collection of natural numbers with an operation of partial application of a number, seen as a program, to another number, seen as an input. A convenient generalization of this notion is that of *partial combinatory algebra*.

Conventions: As usual we denote with $\rightharpoonup$ the partial functional space. $t\downarrow$ denotes the fact that the expression t is defined. $\cong$ is Kleene's equality: $t \cong s$ iff $(t\downarrow \Leftrightarrow s\downarrow) \wedge (t\downarrow \Rightarrow t = s)$, where: $t=s \Rightarrow t\downarrow \wedge s\downarrow$, and $ts\downarrow \Rightarrow t\downarrow \wedge s\downarrow$.

Definition *(partial combinatory algebra)*

A partial combinatory algebra (pca) is a (partial) algebraic structure $(D, k, s, \cdot)$ where: $k, s \in D$, $\cdot : D \times D \rightharpoonup D$. The distinguished elements k, s satisfy the following properties, where we abbreviate $x \cdot y \equiv xy$:

$$kxy = x, \qquad sxy\downarrow, \qquad sxyz \cong xz(yz).$$

Examples *(pcas)*

(1) An important example of pca is *Kleene's* $(\omega, \cdot)$ where ω are the natural numbers and $n \cdot m$ is the n-th Turing machine applied to the input m, namely $n \cdot m = \{n\}m$, given some enumeration $\{\ \}$.

(2) Another canonical example of pca is that of a non-trivial domain D that is a retract of the partial functional space, written $D \rightharpoonup D \lhd D$, in the category of directed complete partial orders and partial continuous maps.

We now have to decide how to interpret formulas and proofs, and more generally types and programs. In Kleene's interpretation, formulas are interpreted as subsets of natural numbers, on the other hand no mention is made of morphisms, hence no obvious interpretation of proofs is available.

The first obvious attempt could consist in interpreting types as subsets of the realizability structure. But in order to have a model of type theory we need at least a ccc, so which are the morphisms? It is clear that to have an interesting structure morphisms have to be somehow "realized". Unfortunately there does not seem to be enough structure to get a ccc. We need a finer description of types. Rather than identifying types with a collection of realizers we consider a type as a partial equivalence relation (per) over the collection of realizers. There is now an obvious notion of morphism that makes the category into a ccc.

Conventions: Supposing A, B,... binary relation over a set D we write: $dAe \equiv_{abr.} (d,e) \in A$, $[d]_A \triangleq \{e \in D \mid dAe\}$, $[A] \triangleq \{[d]_A \mid dAd\}$, $|A| \triangleq \{d \mid d \in D, dAd\}$.

Definition *(partial equivalence relations)*

Let D be a pca. The category of pers over D (per_D) is defined as follows:

$$per_D \triangleq \{A \mid A \subseteq D \times D \text{ and } A \text{ is symmetric and transitive}\}$$
$$per_D[A, B] \triangleq \{f : [A] \to [B] \mid \exists \phi \in D. \forall d \in D.\ dAd \Rightarrow \phi d \in f([d]_A)\}$$

If ϕ is a realizer for the map $f: A \to B$, i.e. $\forall d \in D.\ dAd \Rightarrow \phi d \in f([d]_A)$, we may denote f with $[\phi]_{A \to B}$ (consistently with the definition of exponent in per_D given in the following proposition).

Proposition *(per_D is a ccc)*

The category, per_D, of partial equivalence relations over a pca D is cartesian closed.

Proof

From the operation of application in a pca D one can define, as usual, operations of pairing, $< , >: D\times D\to D$, and projection $\pi_i: D\to D$, (i=1,2) such that $\pi_1<d,d'> = d$, and $\pi_2<d,d'> = d'$. We use λ-terms to represent elements of the pca with a certain functionality. For instance, by $\lambda d.(\pi_1 d)(\pi_2 d)$ we intend an element e of the pca such that, for any d, $ed \cong (\pi_1 d)(\pi_2 d)$. The properties of combinatorial completeness of pcas guarantee that these element exists.

Terminal object. Set $\mathbf{1} \triangleq D\times D$. For $d\in D$ the constant function $\lambda e.d$ realizes the unique map from a per A into **1**.

Product. Define for the product per: $dA\times Be \Leftrightarrow \pi_1 dA\pi_1 e \wedge \pi_2 dB\pi_2 e$. It is immediate to verify that pairing and projections in the pca realize the pairing and projections of the category.

Exponent. Define for the exponent per: $h\ \exp(A, B)\ k \Leftrightarrow \forall d,e.\ dAe \Rightarrow hdBke$. We will abbreviate exp(A, B) with B^A. The evaluation is realized by $\lambda d.(\pi_1 d)(\pi_2 d)$, the natural isomorphism Λ is realized by $\lambda\phi.\lambda c.\lambda a.\phi<a,c>$. □

Exercises: (1) The following remarks should motivate the shift from subsets of D to pers. (a) One can identify the subsets of the realizability structure with those pers that have at most one equivalence relation. Show that the full subcategory composed of these pers is cartesian closed, but each object is either initial or terminal. (b) Consider a category of pers, say **C**, in which the objects are like in (a) but where morphisms are defined as follows: $\mathbf{C}[A, B] \triangleq \{\phi\in D \mid \forall d, e\in D.\ d\in |A| \Rightarrow \phi de\in |B|\}$. Show that **C** is not cartesian closed.

(2) Show that per_D has all finite limits and colimits.

We are now going to give an equivalent presentation of the category of partial equivalence relations, namely the category of *modest (D-)sets*. At the same time we define the bigger category of *D-sets*. This larger category is introduced here just to give a feeling of the fact that pers can be seen as 'set with a realizability relation'. A more satisfactory justification of this statement would require the introduction of the *effective topos* (Hyland[82]) of which both pers and D-sets can be considered as full subcategories. As it will be briefly mentioned later, on D-sets also play an important role in presenting the "internal" completeness properties of pers.

Definition *(D-sets and modest sets)*

A *D-set* is a pair $(X, \Vdash_X)$ where X is a set and $\Vdash_X \subseteq D\times X$ is an *onto* "realizability" relation. A *morphism* of D-sets, say $f: (X, \Vdash_X)\to(Y, \Vdash_Y)$, is a function $f: X\to Y$, such that: $\exists\ \phi\in D.\ \forall d, x.\ (d \Vdash_X x \Rightarrow \phi d \Vdash_Y f(x))$.
A D-set $(X, \Vdash_X)$ is *modest* if $d\Vdash_X x \wedge d\Vdash_X y \Rightarrow x=y$.

Proposition

(1) The category of modest D-sets and Per_D are *equivalent*.

(2) The full subcategory of modest sets is *reflective* in the category of D-sets.

Proof

(1) To the modest set $(X, \Vdash_X)$ associate the per $P(X, \Vdash_X)$ defined as:

$d\, P(X, \Vdash_X)\, e \Leftrightarrow \exists x \in X.\, (d \Vdash_X x \Rightarrow e \Vdash_X x)$.

(2) The basic observation is that if $f: (X, \Vdash_X) \to (Y, \Vdash_Y)$ where $(Y, \Vdash_Y)$ is modest then: $d \Vdash_X x \wedge d \Vdash_X y \Rightarrow f(x)=f(y)$. Because: if f is realized by ϕ then $\phi d \Vdash_Y f(x) \wedge \phi d \Vdash_Y f(y)$, which forces $f(x) = f(y)$. Let us now describe how to associate a modest set to a D-set $(X, \Vdash_X)$. First define a relation R over X as:

$x\, R\, y \Leftrightarrow \exists d \in D.\, (d \Vdash_X x \wedge d \Vdash_X y)$.

Let $\sim$ denote the equivalence relation obtained by the transitive closure of R. Now consider the quotient set $Y = [X]_\sim$ equipped with the relation $\Vdash_Y$ defined as follows:

$d \Vdash_Y [x]_\sim \Leftrightarrow \exists z \in [x]_\sim .\, d \Vdash_X z$.

The structure $(Y, \Vdash_Y)$ is the modest set we looked for. □

2. Interpretation of System F

In this section we introduce the system F and then define its interpretation in the category of pers. System F is the fragment of the calculus introduced in chpt. 6 that does not have dependent types. The calculus is reintroduced here in order to have a more compact notation. Types and raw terms are defined by the following BNFs:

Type Variables:	$tv ::= t \mid s \mid \ldots$
Types:	$\alpha ::= tv \mid (\alpha \to \alpha) \mid (\forall tv.\alpha)$
Term Variables:	$v ::= x \mid y \mid \ldots$
Terms:	$M ::= v \mid (\lambda v{:}\alpha.M) \mid (MM) \mid (\lambda tv.M) \mid (M\alpha)$

A context Γ is formed according to the convention stated in chpt. 2. We denote with $ftv(\Gamma)$ the collection of type variables that occur free in types occurring in Γ. Note that in the calculus presented here (as opposed to the one in chpt. 6) type variables in contexts are left implicit. A *typing judgment* is of the shape $\Gamma \supset M{:}\alpha$ where Γ is always a well formed context. Derivable typing judgments are specified by the following formal system:

(asmp)	$x{:}\alpha \in \Gamma \Rightarrow \Gamma \supset x{:}\alpha$
$(\to I)$	$\Gamma, x{:}\alpha \supset M{:}\beta \Rightarrow \Gamma \supset (\lambda x{:}\alpha.M){:}(\alpha \to \beta)$
$(\to E)$	$\Gamma \supset M{:}(\alpha \to \beta)$, $\Gamma \supset N{:}\alpha \Rightarrow \Gamma \supset (MN){:}\beta$
$(\forall I)$	$\Gamma \supset M{:}\alpha \; t \notin ftv(\Gamma) \Rightarrow \Gamma \supset (\lambda t.M){:}(\forall t.\alpha)$
$(\forall E)$	$\Gamma \supset M{:}(\forall t.\alpha) \Rightarrow \Gamma \supset (M\beta){:}[\beta/t]\alpha$

Interpretation

Since perD is a ccc we already know how to interpret the simply typed fragment of system F. On the other hand the interpretation of the clauses $(\forall I)$ and $(\forall E)$ is more problematic. In principle we could introduce a general definition of categorical model of system F and show that per_D satisfies all requirements. This

however would represent too long a detour. We shall limit ourselves to the definition of an interpretation. In order to show that the interpretation is well defined we introduce an auxiliary interpretation of the 'type-free terms' which allows to express a certain 'uniformity' of the main interpretation w.r.t. type abstraction and type application.

• **Types**. Given a type assignment η: tv$\to$per$_D$, the interpretation of a type is a per defined by induction as follows:

$$[\![t]\!]\eta = \eta(t)$$
$$[\![\alpha\to\beta]\!]\eta = \exp([\![\alpha]\!]\eta, [\![\beta]\!]\eta)$$
$$[\![\forall t.\alpha]\!]\eta = \cap_{A\ per}[\![\alpha]\!]\eta[A/t]$$

If $\Gamma \equiv x_1: \alpha_1,...,x_n: \alpha_n$ then $[\![\Gamma]\!]\eta = (..(\mathbf{1}\times A_1)\times...\times A_n)$, where: $A_i = [\![\alpha_i]\!]\eta$, i=1,..,n.

• **Type Free Terms**. The function *er* (erasure) takes a typed term and returns a type-free term in the language generated by the following BNF (as usual v stands for the variables):

$$P ::= v \mid (\lambda v.P) \mid (PP)$$

It is defined by induction on the structure as follows:

$$er(x) = x;\ er(\lambda x{:}\alpha.M) = (\lambda x.er(M));\ er(MN) = (er(M)er(N));$$
$$er(\lambda t.M) = er(M);\ er(M\beta) = er(M).$$

If $\vdash\Gamma \supset M: \alpha$ then $[\![\Gamma\supset er(M): \alpha]\!]\in D$, and it is defined by induction on the length of the typing proof as follows:

(asmp) $[\![\Gamma \supset x_i: \alpha_i]\!] = p_{n,i}$

($\to$I) $[\![\Gamma \supset \lambda x.er(M):\alpha\to\beta]\!] = \lambda c.\lambda a.([\![\Gamma,x:\alpha \supset er(M):\beta]\!] \langle c,a\rangle)$

($\to$E) $[\![\Gamma\supset er(M)er(N):\beta]\!] = \lambda c.([\![\Gamma\supset er(M):\alpha\to\beta]\!] c)([\![\Gamma\supset er(N):\alpha]\!] c)$

($\forall$I) $[\![\Gamma\supset er(M):\forall t.\alpha]\!] = [\![\Gamma\supset er(M): \alpha]\!]$

($\forall$E) $[\![\Gamma\supset er(M): [\beta/t]\alpha]\!] = [\![\Gamma\supset er(M): \forall t.\alpha]\!]$

where: (i) as usual $p_{n,i}$ is a suitable combination of the realizers for projections, (ii) $\langle , \rangle$ is the coding for the pairs, (iii) the λ-terms have to be compiled in the language of combinators in order to simulate λ-abstraction.

Proposition

Suppose $\vdash\Gamma \supset M: \alpha$, then for any type assignment η we have:

$[\![\Gamma\supset er(M):\alpha]\!] \in \exp([\![\Gamma]\!]\eta, [\![\alpha]\!]\eta)$.

Proof

This is a simple induction on the length of the typing judgment. Observe that the interpretation of (asmp) has been devised in order to obtain as result a realizer for the corresponding 'typed' projection. The interpretations for ($\to$I) and ($\to$E) just use the realizers of the natural transformation Λ and of the evaluation. The crucial

point is (∀I) where we use the side condition t∉ftv(Γ). (∀E) follows by the interpretation of second order quantification as intersection. □

• **Typed Terms**. Given a type assignment η the interpretation of a judgment ⊢Γ⊃M:α is a morphism in per from ⟦Γ⟧η to ⟦α⟧η that is defined as follows:

$$⟦Γ ⊃ M:α⟧η = [\, ⟦Γ⊃er(M):α⟧ \,]_{exp(⟦Γ⟧η, ⟦α⟧η)}.$$

This is the unique morphism from ⟦Γ⟧η to ⟦α⟧η that is realized by the interpretation of the erased judgment.

3. Towards Recursion

In the remaining part of this chapter we concentrate on the problem of giving a per interpretation of type theories including recursion. As usual one is naturally led towards a notion of 'complete' partially ordered set.

At the same time we want to stay faithful to our goal of regarding data types as particular sets of our realizable universe. Hence we look for certain sets on which it is possible to find an *"intrinsic order"* in such a way that *all* set-theoretic maps *continuously preserve* this intrinsic order. The method will be that of restricting the attention to a *full, reflective* subcategory of pers.[33]

In this section we develop a notion of *intrinsic pre-order* working in an arbitrary *partial cartesian closed category* (pccc). Every pccc has an object Σ that classifies the *admissible monos* and that can be regarded as an abstraction of Sierpinski space (cf. chpt. 4). We recall that the following isomorphism holds: a→Σ ≅ a⇀1;[34] hence given f: a→Σ and x: 1→a we have an induced notion of *convergence* for fx. Given two points of an object a, say x,y: 1→a, x is *intrinsically less* than y if any map f: a→Σ that converges on x also converges on y. Maps automatically preserve the intrinsic pre-order. A separated object is an object in which the intrinsic pre-order is a partial order (i.e. anti-symmetry holds).

In section 4 we restrict our attention to a pccc of *pers over an arbitrary pca*. The convergence predicate of the pca naturally induces on any given per a family of *semi-computable* predicates, that, on the other hand, gives rise to a family of *admissible monos* generating the pccc. *Separated pers* (Σper) are those pers whose points are separated by the topology induced by the semi-computable subsets. They are shown to form a full reflective sub-category of per and therefore they enjoy all the completeness properties of the larger category. The category of separated pers can be seen as a universe of partially ordered sets and monotone maps.

In section 5 the notion of *N-completeness* is introduced. Σper contains a natural number object (N). Given a Σper A we will say that A is N-complete if every chain (w.r.t. the intrinsic order) described by a map from N to A has a

[33] We emphasize that the following sections are just an introduction to a quickly developing area.
[34] a⇀1 is the set of "partial" maps from a into the terminal object.

supremum in A. When working on Kleene's pca $(\omega, \cdot)$ it will turn out that all realized maps between N-complete Σper ($c_N\Sigma$per) are continuous in the sense of preserving suprema of these chains.

Intrinsic Pre-order

In chapter 4 we have introduced some basic notions about *partial cartesian closed categories* (pccc) and their properties. Every pccc has an object Σ, called *dominance*, that classifies the *admissible subobjects* in the same sense as the object of truth-values Ω classifies arbitrary subobjects in a topos.

The object Σ induces a *preorder* $\leq_a$ on the points of every object a (i.e. maps from the terminal object into a). Following Rosolini[86] we focus on the full subcategory of *separated objects*, that is composed of those objects for which $\leq_a$ is antisymmetric.

In a pccc the maps from an object a to the dominance Σ play the role of convergence tests. These tests induce a pre-order on the points of an object. The idea of ordering points by tests bears a striking analogy with the one encountered in, so-called, operational semantics of ordering terms by observations. However a crucial difference is that in our case the pre-order does not depend on the language at hand but only on a global category on which one can interpret a variety of constructs.

Convention *(Points and convergence predicate)*

We write x: a to indicate that x is a *point* of a, that is a morphism $x:1\to a$. Since we will be dealing with (p)ccc we confuse points in the objects $a\to b$ and $a\rightharpoonup b$ with morphisms, respectively, in C[a,b] and pC[a, b]. E.g. $f: a\to b$ can be seen both as a morphism from a to b and as a point in $a\to b$.

We introduce a *convergence predicate*, $\downarrow$, as follows: if $x:a$, $p\equiv[m, f]: a\rightharpoonup b$, with $m:d\rightarrowtail a$, $f:d\to b$ then we write: $px\downarrow \Leftrightarrow \exists\, h: 1\to d.\ (mh = x)$. By virtue of a well-known isomorphism we also write both $p: a\rightharpoonup b$ and $p: a\to(b)_\perp$. The reader will have noticed our last abuse: sometimes we omit to write composition.

Definition *(Intrinsic preorder)*[35]

Let (C, M) be a pccc and a an object. Define a preorder $\leq_a$ on the points of a as:

$$\text{if } x, y: a \text{ then } x\leq_a y \text{ iff } \forall p:a\to\Sigma.(px\downarrow \Rightarrow py\downarrow).$$

The intuition is that x is less then y in a if every convergence test $p:a\to\Sigma$ that succeeds on x also succeeds on y. In the following we also write $px \preccurlyeq py$ for $px\downarrow\Rightarrow py\downarrow$ and $px\equiv py$ for $px\downarrow \Leftrightarrow py\downarrow$.

[35] Every partial category pC has a partial ordering on morphisms given by the containment of the convergence spaces. In such partial ordering the total morphisms represent the maximal elements. Therefore the points of an object a, i.e. the total maps $x:1\to a$, turn out to be all incomparable. On the other hand the intrinsic preorder may give non-trivial partial orders on the points.

Definition *(category of Σ-objects)*

Given a pccc (C, M) with dominance Σ we denote with ΣC the full subcategory of C whose objects enjoy the property that the intrinsic preorder is anti-symmetric. An object a such that $\leq_a$ is a partial order is called *Σ-object* or, equivalently, *separated object.*

Proposition *(Basic properties)*

Let (C, M) be a pccc with dominance Σ. Then:

(1) Maps preserve the intrinsic preorder.

(2) Σ-objects are closed under subobjects.

(3) Moreover if (C, M) has enough points (i.e. 1 is a generator) and a is an object then Σ^a is a Σ-object.

Proof

(1) Let f: a→b and x, y: a. Suppose $x\leq_a y$, then given any p: b→Σ we have by hypothesis pfx≤pfy, since pf: a→Σ. Hence $fx\leq_b fy$: b.

(2) Let m:a↣b be a mono and b be a Σ-object. If x and y are two distinct points in a then mx and my are two distinct points in b. Hence, since b is a Σ-object, they are separable by a map p: b→Σ. Then the map pm separates the points x and y.

(3) If f, g:Σ^a and f≠g then, by the enough point assumption, there is a x:a s.t. ~fx≡gx. Take λh:Σ^a.hx: Σ^a→Σ as separator for f and g. □

4. Separated Partial Equivalence Relations

One can build over every partial combinatory algebra (pca) D the category of partial equivalence relations, per_D, and a family of admissible monos, M_D. The partial category (per_D, M_D) turns out to be a *pccc with enough points.* Besides, Σper_D, the category of separated pers over D, is *reflective* in per_D.

Semi-computable predicates on D

Partiality is explicitly given in D and by generalizing basic facts of recursion theory (i.e. r.e. sets are exactly the domain of computable functions) it also provides a notion of *semi-computable predicate on D.*

(1) $\Sigma(D) \triangleq \{W \mid W \subseteq D, \exists\, d\in D\ \ W = Dom(d)\}$, where $Dom(d) \triangleq \{e \mid e\in D\,,\ de\downarrow\}$.

(2) The collection of predicates Σ(D) induces a *refinement pre-order* on D defined as: $d\leq_D e$ iff $\forall W\in\Sigma(D).(d\in W \Rightarrow e\in W)$

(3) Observe that the operation of application preserves this pre-order:
$\forall e\in D.(d\leq_D d' \Rightarrow ed\leq_D ed')$.

(4) If the pca is not total then Σ(D) can be seen as a *basis for a topology* as:
(a) ∅, D∈Σ(D). (b) If W, W'∈Σ(D) then W∩W'∈Σ(D). Because:
(a) Take respectively the always divergent and always convergent map.
(b) If W=Dom(e) and W'=Dom(e') then

$$W\cap W' = Dom(\lambda d.\text{if } (ed\downarrow \wedge e'd\downarrow) \text{ then}\downarrow) = Dom(\lambda d.(\lambda x.\lambda y.c)(ed)(e'd)) \ \square$$

PER as a Pccc

We show that given any per, A, $\Sigma(D)$ induces a collection, $\Sigma(A)$, of semi-computable predicates on A. From this structure it is easy to obtain a family M_D of admissible monos on per_D that turns the category into a pccc.

Definition *(Semi-computable subsets of a per)*

Let $A \in per_D$, define:

$$\Sigma(A) \triangleq \{B \in per_D \mid [B] \subseteq [A], \exists W \in \Sigma(D).|A| \cap W = |B|\}$$

In other words B belongs to $\Sigma(A)$ if the equivalence classes in B form a subset of those in A and there is a set $W \in \Sigma(D)$ that separates B from the other equivalence classes in [A].

Closure properties of $\Sigma(A)$

Let D be a non-total pca. Then $\Sigma(A)$ enjoys closure properties analogous to $\Sigma(D)$:

(a) $\emptyset, A \in \Sigma(A)$. (b) If $B, B' \in \Sigma(A)$ then $B'' \in \Sigma(A)$

where B'' is the per corresponding to the partial partition $[B]\cap[B']$.

Definition *(Admissible family of monos)*

Define M_D as the following family of monos:

$m: A' \rightarrowtail A \in M_D(A)$ iff $A' \in \Sigma(A)$ and m is the inclusion morphism.

Note that the morphism m is realized by the identity. It is easy to check that this collection of monos is indeed admissible. The conditions for identity and composition are clear. Let us consider the case for the pullbacks: assume $f:A\to B$ and $m:C\rightarrowtail B$ with ϕ realizer of f and $|B|\cap Dom(\psi) = |C|$. To construct the pullback consider $W' = Dom(\lambda d.\psi(\phi d))$ and the related C' admissible subobject of A.

The category (per_D, M_D) of pers and partial maps is equivalent to the category $pper_D$ defined as follows:

$$Ob_{pperD} \triangleq Ob_{perD}$$

$$pper_D[A, B] \triangleq$$

$$\triangleq \{f:[A]\rightharpoonup[B] \mid \exists \phi \in D.\forall d.dAd \Rightarrow ((\phi d\downarrow \Leftrightarrow f([d]_A)\downarrow) \wedge (\phi d\downarrow \Rightarrow \phi d \in f([d]_A))\}$$

Proposition *(pper is a pccc)*

The category (per_D, M_D) is a *pccc*. In particular the *partial exponent* is defined as: $f(A\rightharpoonup B)g \Leftrightarrow \forall d, e.(dAe \Rightarrow fd \cong_B ge)$, where $\cong_B$ is Kleene equality relativized to B, namely: $t \cong_B s \Leftrightarrow (t\downarrow \Leftrightarrow s\downarrow)$ and $(t\downarrow \Rightarrow tBs)$.

Notice that the category has *enough points*. The *terminal object* is any per with one equivalence class, say $1 \triangleq D\times D$. In this case the *dominance* is $\Sigma \triangleq 1\rightharpoonup 1=\{\bot, \top\}$,

where: $\bot \triangleq \{d \in D \mid \forall e.de\uparrow\}$, $\top \triangleq \{d \in D \mid \forall e.de\downarrow\}$.

D-order implies intrinsic order on pers

Applying the previous construction we can now define:

Σper_D as the full subcategory of per_D whose objects are Σ-objects.

Note: Observe that if $d \leq_D d'$, $A \in per_D$, and $d, d' \in |A|$ then a fortiori $[d]_A \leq_A [d']_A$. Because: suppose $B \in \Sigma(A)$ and $[d]_A \in [B]$, then there is a $W \in \Sigma(D)$ such that $|A| \cap W = |B|$. But by hypothesis $d' \in W$ and therefore $[d']_A \in [B]$.

The sense of this note is that if two elements cannot be separated in the type free universe of the realizability structure D then a fortiori they cannot be separated in the "typed" structure of pers. □

We now give an elementary and direct proof that Σper_D is reflective in per_D.

Theorem *($\Sigma per \subset_{>} per$)*

Σper is a full reflective subcategory of per.

Proof

The very simple idea for obtaining a Σper $L_\Sigma(A)$ from the per A is to collapse equivalence classes that cannot be separated by $\Sigma(A)$. Given a per A and the intrinsic preorder $\leq_A$ we can define, as usual, an equivalence relation, $\approxeq_A$, on [A] as: $[d]_A \approxeq_A [e]_A \Leftrightarrow d, e \in |A| \wedge [d]_A \leq_A [e]_A \wedge [e]_A \leq_A [d]_A$.

Now define the reflector L_Σ: per$\rightarrow\Sigma$per as follows:

if $A \in per$ then $L_\Sigma(A)$ is such that: $dL_\Sigma(A)e \Leftrightarrow [d]_A \approxeq_A [e]_A$

if $f: A \rightarrow B$ then $L_\Sigma(f)([d]_{L\Sigma(A)}) = f([d]_A)$

One can easily verify that: (1) $L_\Sigma(A)$ is a Σper. Actually it is the least Σper containing A (as a relation). (2) $dL_\Sigma(A)e \Rightarrow f([d]_A) = f([e]_A)$, as B is separated. (3) Every map from a per A to a Σper B can be uniquely extended to a map from $L_\Sigma(A)$ to B. From these facts it is easy to exhibit the natural isomorphism of the adjunction. □

The following corollary resumes our progress: we have managed to build a full sub-category of pers that has the same closure properties of pers, and moreover has an intrinsic notion of partial order that will turn out to be useful in the interpretation of recursion.

Corollary *(Closure properties of Σper)*

Σper is a ccc and it has all limits and colimits of per.

Proof

The existence of limits and colimits is guaranteed by the reflection. Let us check that Σper is closed under the usual definition of exponent in per. Suppose $B \in \Sigma\text{per}$ and f, g: $A \to B$. Suppose that f and g are distinct, then there is a point x: A such that fx, gx are distinct and, by hypothesis, separable by means of k: $B \to \Sigma$. Then the map λh: $A \to B.k(hx)$ separates f and g. □

5. N-completeness

We are interested in finding an analogous of the notion of ω-completeness in a realizability framework. In the first place we need an object N that can play the role of the *natural numbers*.[36]

Natural number object in a generic pca.

Denote with $\underline{n}$ the term corresponding to the numeral n in a pca, say the compilation in a pca of the n-th Church numeral. Let n^D be the interpretation of the numeral $\underline{n}$ in the pca D. We set:

$$N_{CN} \triangleq \{ \{ n^D \} \mid n \in \omega \}$$

In a pca all the basic arithmetic operations are definable on the numerals. As a consequence of this fact if D is non-trivial and $n \neq m$ then $n^D \neq m^D$. Moreover $N_{CN} \in \Sigma\text{per}$. Finally one can show that N_{CN} is a nno in the category. □

Natural number object in $(\omega,\cdot)$.

When working in $(\omega,\cdot)$ we can define as nno:

$$N_\omega \triangleq \{ \{ n \} \mid n \in \omega \}$$

In the following we will use the fact that $\{\{n\} \mid nn\uparrow\}$ does not belong to $\Sigma(N)$. In a generic pca having all the tools of arithmetic we can still express concepts like "the n-th turing machine applied at input m converges in at least k-steps" however the problem is that it can still be the case that $\{\{n\} \mid nn\uparrow\}$ belongs to $\Sigma(N)$. This is not surprising as the operation of application can be completely unrelated to the gödel-numbering. For example consider the pca satisfying in the category of directed complete partial orders and partial continuous maps: $\omega + (D \to D) \triangleleft D$, where ω are the natural numbers with the flat ordering. Then $N_D \triangleq \{ \{ d \} \mid d$ is in the image of ω via embedding$\}$ is a nno, however all its subsets are still semi-computable as they are Scott open. □

[36] More precisely a natural number object (nno) is a diagram $1 \to^0 N \to^s N$ that is initial among all diagrams of the shape: $1 \to^x A \to^f A$.

N-completeness

Next we will concentrate on the *N-complete Σpers* that is those Σpers such that any ascending sequence on them, that is definable as a map in the category, has a lub w.r.t. the intrinsic order.

When restricting the attention to *N-complete Σper over Kleene's pca* it is possible to prove a remarkable generalization[37] of Myhill-Shepherdson's theorem (see chpt. 1) asserting that all the maps preserve lubs of chains definable in the category. This will arise as a corollary of the fact that for any per A the elements of $\Sigma(A)$ are (N-)Scott's opens.

A remarkable application of this fact is that the full subcategory of N-complete, separated pers can be seen as a sort of *pre-O-category* in that the morphisms are partially ordered, there are lubs' of (N-)chains, and the operation of composition preserves this structure. When stating the completeness condition for a Σper A we will only be interested in the *existence* of the least upper bounds (lubs) of the chains, $\chi:N\to A$, that are definable as maps from the nno N to A, briefly we will write

$$\chi:AS(A) \text{ (AS for ascending sequence), if } \chi: N\to A \text{ and } \forall n:N.(\chi n \leq_A \chi n+1).$$

Observe that whenever we select a subset of the equivalence classes of a (Σ)per we can naturally consider it as a (Σ)per. For example AS(A) is a subset of $[N\to A]$ and $U\in\Sigma(A)$ is a subset of [A]. According to a *constructive* reading the existence of the lub of every ascending sequence implies the existence of a method to find this lub given a realizer for the sequence. Indeed as soon as one considers the problem of the closure of the collection of N-complete objects w.r.t. the functional space constructor it is noticed that it is important to have a realizer that *uniformly*, for every ascending sequence of a given type, computes the lub (we refer to Phoa[90], Freyd&al.[90], Amadio[90] for more information on the closure properties of this category). This motivates the following definition.

Definition *(N-completeness)*

A Σper A is N-complete if $\forall\chi:AS(A).\ \exists\ \mathrm{lub}_A\chi$,

where *the existence of lub has to be interpreted constructively*, that is:

(a) $\forall\chi:AS(A).\exists x:A.(\forall n:N.(\chi n\leq_A x)\ \wedge\ \forall y:A.(\forall n:N.(\chi n\leq_A y)\Rightarrow x\leq_A y)$ and

(b) $\exists\sigma_A{:}\ AS(A)\to A.\forall\chi:AS(A).\sigma_A(\chi) = \mathrm{lub}_A\chi$.

Since the map σ_A, if it exists, is uniquely determined we will simply indicate with A rather than with (A, σ_A) an N-complete Σper (henceforth $c_N\Sigma$per).

Note: Given N it is fairly easy to show that $c_N\Sigma$per is a non-trivial category. For example every separated object A in which all elements are incomparable is going to be in $c_N\Sigma$per as one can define $\sigma_A \triangleq \lambda\chi:AS(A).\ \chi(0)$ where 0 is the zero of the nno (every ascending sequence on a flat object is constant).

[37] See note 3.2.6.(1) for a rephrasing of the original statement of the theorem.

Definition *(Scott open)*

Let A be a per. A subset U of [A] is a Scott open and we write $U\in\tau(A)$ iff
(1) $\forall x,y:A.(x:U\wedge x\leq_A y \Rightarrow y:U)$. (2) $\forall\chi:AS(A).(\exists\, lub_A\chi:U \Rightarrow \exists n:N.(\chi n:U))$.
Note that this definition makes sense in any preorder.

Realized maps are N-continuous

It is immediate to check that $\tau(A)$ defines a topology over [A]. Before stating the main result we need to look closer at the nno.

Theorem *($\Sigma(A)\subseteq\tau(a)$)*

If $A\in c_N\Sigma per_\omega$ and $U\in\Sigma(A)$ then $U\in\tau(A)$.

Proof

The first condition of upward closure follows by the definition of intrinsic order. Take x:U and suppose $x\leq_A y$ then y:U as $x\leq_A y$ iff $\forall U\in\Sigma(A).(x:U\Rightarrow y:U)$.

The proof of the second condition takes advantage of the specific recursion-theoretic character of the structures we have built. In this sense the following proof is the kernel of this approach to the development of the theory. In the following: $N\equiv N_\omega$.

Consider the set $K\triangleq\{\{n\}\mid nn\downarrow\}\subseteq[N]$. Observe $K\in\Sigma(N)$ and $K^c\notin\Sigma(N)$, where K^c is K complement w.r.t. N. Consider the predicate $nn\downarrow i$ meaning that the computation nn of the n-th machine applied to the input n will stop in at most i steps. This is a decidable predicate.

Now let us proceed *by contradiction* assuming there is χ: AS(A) such that:

$\exists\, lub_A\chi$: U and $\forall n:N.\ \sim(\chi n:U)$.

The crucial idea is to build a map h: $N\to A$ such that, under the condition $\forall n:N.\ \sim(\chi n:U)$, maps K in $\{\chi n\mid n:N\}$ and K^c in $lub_A\chi$. Next one uses the pullback condition to conclude that $h^{-1}(U)=K^c\in\Sigma(N)$.

We now define a family of chains $c(n,i): N\to(N\to A)$. Let $\mu k\leq i.nn\downarrow k$ be the least element k less then i s.t. nn converges in k steps. Define

$$c(n,i)\triangleq \text{if } \sim(nn\downarrow i) \text{ then } \chi i \text{ else } \chi(\mu k\leq i.nn\downarrow k)$$

Observe that for any given n if $n\in K$ then c(n,i) coincides with the ascending sequence χ up to the first k s.t. $nn\downarrow k$ and then becomes definitely constant; on the other hand if $n\in K^c$ then c(n,i) coincides with χ.

We use the existence of a map that uniformly realizes the lub of ascending sequences to define the map $h:N\to A$ as follows:

$$h(n)\triangleq\sigma_A(\lambda i.c(n,i)).$$

As we have just observed $h(n)=lub_A\chi$ if $n\in K^c$ and $h(n)\in\{\chi n\mid n:N\}$ o.w..
From this we can conclude $h^{-1}(U)=K^c\in\Sigma(N)$ obtaining the desired contradiction. □

For the logically inclined reader we mention that this proof by contradiction can be turned into a "constructive proof" via Markov Principle.

Definition *(preservation of N-chains)*

Let A, B$\in \Sigma$per. Then we say that f:A$\to$B *preserves N-chains* iff

$\forall\chi$: AS(A). ($\exists$ lub$_A\chi$:A $\Rightarrow$ ($\exists$ lub$_B$fχ:B $\wedge$ f(lub$_A\chi$)=lub$_B$fχ)).

Proposition *(Realized maps are Scott continuous)*

Suppose A, B$\in c_N\Sigma per_\omega$.

(1) Any map f: A$\to$B preserves N-chains.

(2) If f: A$\to$B preserves N-chains then it is (N-)Scott continuous.

Proof

(1) Consider χ: AS(A) and assume $\exists$lub$_A\chi$:A. In order to show $\exists$lub$_B$fχ = f(lub$_A\chi$) we prove that for any majorant y:B of fχ:AS(B) we have f(lub$_A\chi$)$\leq_B$y. We recall that f(lub$_A\chi$)$\leq_B$y iff $\forall U\in\Sigma(B)$.(f(lub$_A\chi$):U $\Rightarrow$ y:U). Now U$\in\Sigma$(B) implies by the pullback condition of admissible domains: $f^{-1}(U)\in\Sigma(A)$ i.e. by the previous theorem $f^{-1}(U)\in\tau(A)$. Since f(lub$_A\chi$):U we have lub$_A\chi$:f^{-1}(U) that implies by the definition of open set $\exists$n:N.(χn:f^{-1}(U)). Therefore $\exists$n:N.(fχn:U) and this implies y:U.

(2) As usual take U$\in\tau$(B) and consider f^{-1}(U). This is upward closed by the fact that f is monotone. Moreover consider χ: AS(A) and suppose $\exists$ lub$_A\chi$:f^{-1}(U). Then by hypothesis f(lub$_A\chi$)=lub$_B$fχ : U. Therefore $\exists$n:N.(fχn:U) i.e. χn:f^{-1}(U). □

Note *(On Myhill and Shepherdson theorem)*

Suppose $\{E_n\}_{n\in\omega}$ is an enumeration of finite subsets of ω and < , >: $\omega\times\omega\to\omega$ is a pairing map. Given X, Y$\in 2^\omega$ define:

$$X\cdot Y \triangleq \{m \mid \exists n.(\langle n, m\rangle\in X \wedge E_n\subseteq Y)\}$$

The reader may recognize in this definition the notion of application that arises in graph models. Denote with RE the collection of recursively enumerable sets. Every W$\in$RE determines a functional F_W: RE$\to$RE that is defined as:

$$F_W(X) \triangleq W\cdot X$$

These functionals are called *enumeration operators* (or also recursive functionals). It is immediate to check that these functionals are continuous w.r.t. the order given by set-containment.

Next observe that RE can be seen as a totally enumerated set and therefore as a total equivalence relation. In particular it can be identified with $Id_\omega\to 1$ where: nId_ωm $\Leftrightarrow$ n=m and $\forall$n,m. n1m. Note that the intrinsic order on $Id_\omega\to 1$ coincides with the one given by set-containment.

Myhill-Shepherdson theorem can then be stated as follows:

$$F\in per_\omega[RE, RE] \Leftrightarrow F \text{ is an enumeration operator}$$

This was intended as a characterization of certain "type-2" functionals which are called *effective operations* and which correspond exactly to the morphisms in per_ω[RE, RE]. The interesting part of the proof consists in showing that the effective operations are enumeration operators and, therefore, are continuous. This is the sense that is generalized above.

Bibliography

Abramsky S. [1991] "Domain theory in logical form", APAL 51.

Abramsky S. [1990] "The lazy lambda calculus", In D. Turner (ed.), Research topics in functional programming, Addison-Wesley.

Abramsky S. [1991] "A domain equation for bisimulation", Information and Computation, 92, (161-218).

Aczel P. [1988] "Non-well founded sets", CSLI Lecture Notes 14.

Amadio R. [1988] "Recursion over realizability structures", Information&Computation, 91, 1, (55-85), preliminary version appeared as TR 1/89 Dipartimento di Informatica, Università di Pisa.

Amadio R. [1990] "Domains in a Realizability Framework", in Proc. CAAP91, Brighton, SLNCS 493, Abramsky S., Maibaum T. (eds.), (241-263). Extended version appeared as TR 19-90, Liens Paris.

Amadio R. [1991] "Bifinite Domains: Stable Case", in Proc. Category Theory in Comp. Sci. 91, Paris, Pitt&al. (eds.), SLNCS 530. Full version appeared as TR 3-91, Liens Paris.

Amadio R. [1992] "A quick construction of a retraction of all retractions for stable bifinites", Info.&Comp., to appear. Also appeared as RR 1785, INRIA-Lorraine.

Amadio R., Bruce K., Longo G. [1986] "The finitary projection model and the solution of higher-order domain equations", in Proc. IEEE Conference on Logic in Computer Science (LICS), Boston, June 1986, (122-130).

Amadio R., Cardelli L. [1990] "Subtyping Recursive Types", ACM-TOPLAS, 15,4, (575-631), 1993. Extended abstract in Proc. ACM-POPL91, Orlando.

Amadio R., Longo G. [1986] "Type-free compiling of parametric types", presented at the IFIP Conference on Formal Description of Programming Concepts, Ebberup (DK), published in Formal Description of Programming Concepts-III, M. Wirsing (Ed.), North Holland, 1987 (377-398).

Asperti A., Longo G. [1991] "Categories, Types, and Structures", MIT-Press.

Barendregt H. [1984] "The lambda calculus; its syntax and semantics", Revised and expanded edition, North Holland.

Barr, M. Wells, C. [1985] "Toposes, Triples and Theories" Gundlehren der mathematischen Wissenschaften, 273, Springer-Verlag, New York.

Barr, M. Wells, C. [1990] "Category Theory for Computing Science", Prentice-Hall.

Berardi S. [1991] "Retractions on dI-domains as a model for Type:Type", Info.&Comp., 94, (204-231).

Bérnabou, J. [1985] "Fibered Categories and the Foundations of Naive Category Theory", J. Symbolic Logic, 50, 1, (10-37).

Berry G. [1978] "Stable models of typed λ–calculi", Proc. 5th ICALP, LNCS 62, (72-89).

Berry G. [1979] "Modèles complètament adéquats et stables des lambda-calculs typés", Thèse d' Etat, Université Paris VII.

Berry G., Curien P.L. [1982] "Sequential Algorithms on Concrete Data Structures", Theor.

Comp. Sci., 20, (265-321).

Böhm C., Berarducci A. [1985] "Automatic synthesis of typed lambda-programs on term algebras", Theor. Comp. Sci., 39, (135-154).

Bruce K., Longo G. [1990] "A modest model of records, inheritance and bounded quantification", Info&Comp., 87, 1-2, (196-240), Preliminary Version in Proc. of IEEE-LICS 88.

Church A.[1940] "A formalisation of the simple theory of types", JSL 5, (56-58).

Coquand T., Huet G. [1988] "A calculus of constructions", in Info.&Comp., 76.

Coquand T. [1989] "Categories of embeddings", TCS, 68, (221-237).

Coquand T., Gunter C.,Winskel G.[1988] "Domain theoretic models of polymorphism", Info& Comp. , 81, (123-167).

Curien P.L. [1986] "Categorical Combinators, Sequential Algorithms and Functional Programming" Research Notes in Theoretical Computer Science. Pitman, London. Revised edition, Birkhäuser, 1993.

Curien P-L., Obtulowicz A. [1988] "Partiality, Cartesian Closedness and Toposes", Information & Computation, 80, (50-95).

Curry H. B., Feys R. [1958] "Combinatory Logic, Vol. I.", North-Holland.

Curry H. B., Hindley J. R., Seldin J. [1972] "Combinatory Logic, vol. II", North-Holland.

Cutland N.J. [1980] "Computability, An Introduction to recursive function theory", Cambridge University Press.

de Bruijn, N. G. [1980] "A survey of the project AUTOMATH," in Hindley and Seldin[1980], (589-606).

Danvy O., Filinski A. [1992] "Representing control: a study of the CPS transformations", Math. Str. in C.S., 2, (361-391).

Dilworth, Gleason [1962] "A generalized Cantor theorem", in Proc. Amer. Math. Soc., 13.

Droste M., Göbel R. [1990] "Universal domains in the theory of denotational semantics of programming languages", IEEE-LICS 90, Philadelphia.

Engeler E. [1981] "Algebras and Combinators", Algebra Universalis, 13, (389-392).

Ershov, Ju. L. [1976] "Model C of the partial continuous functionals," Logic Colloquium 76 (Gandy, Hyland eds.), North Holland, 1977.

Felleisen M., Friedman D., Kohlbecker E., Duba B. [1987] "A syntactic theory of sequential control", Theor. Comp. Sci., 52, 3, (205-237).

Freyd P., Mulry P., Rosolini G., Scott D. [1990] "Extensional Pers", 5th IEEE-LICS, Philadelphia.

Gierz G., Hofmann K.H. Keimel K., Lawson J.D., Mislove M., Scott D.S. [1980] "A compendium of continuous lattices", Springer-Verlag.

Girard J.Y.[1986] "The system F of variable types, fifteen years later", Theor. Comp. Sci., 45, (159-192).

Girard J.Y.[1987] "Linear Logic", Theor. Comp. Sci., 50, (1-102).

Girard J.Y., Lafont Y., Taylor P. [1989] "Proofs and Types", Cambridge University Press.

Goguen J. [1991] "A categorical manifesto", in MSCS.

Griffin T. [1990] "A formulae-as-types notion of control", in Proc POPL 90, San Francisco.

Gunter C. [1985] "Universal profinite domains", Info.&Comp., 72, (1-30).

Gunter C., Jung A. [1990] "Coherence and Consistency in Domains", J.P.Apl.Alg., 63, (49-66).

Harper R., Honsell F., Plotkin G. [1987] "A framework for defining logics", LICS 87, Cornell.

Hennessy M., Plotkin G. [1979] "Full abstraction for a simple parallel programming language", in MFCS, SLNCS 74, (108-120).

Henkin L. [1950] "Completeness in the theory of types." JSL 15, (81-91).

Hindley R., Longo G. [1980] "Lambda-calculus models and extensionality", Zeit. Math. Logik Grund. Math. 26, 2 (289-310).

Hindley R., Seldin, J. (eds.) [1980] To H.B. Curry: Essays in Combinatory Logic, Lambda calculus and formalism, Academic Press.

Hindley R., Seldin J. [1986] Introduction to Combinators and Lambda-Calculus, London Mathematical Society.

Howard W. [1980] "The formulas-as-types notion of construction", in Hindley and Seldin 1980, (479-490). (Manuscript written in 1969).

Hyland M. [1976] "A syntactic characterization of the equality in some models of lambda calculus" J. London Math. Soc. 2, 12.

Hyland M. [1982] "The effective Topos", in The Brouwer Symposium, (Troelstra, Van Dalen eds.) North-Holland.

Hyland M. [1988] "A small complete category", APAL, 40, 2, (135-165).

Hyland M. [1991] "First steps in synthetic domain theory", in Proc. Category Theory 90, Carboni&al. (eds.), Springer-Verlag.

Jacobs B., Moggi E., Streicher T. [1991] "Relating models of impredicative type theories", in Proc. CTCS 91, Paris, Pitt D. (ed.), SLNCS 530, (197-218).

Jech T. [1978] "Set Theory", Academic Press.

Johnstone P. T. [1982] "Stone Spaces", Cambridge Studies in Adv. Math. 3.

Jung A. [1988] "Cartesian closed categories of domains", CWI Tracts, Amsterdam.

Jung A. [1990] "Cartesian closed categories of algebraic cpos", TCS, 70, (233-250).

Jung A. [1990] "The classification of continuous domains", IEEE-LICS 90, Philadelphia.

Jung A. [1991] "The dependent product construction in various categories of domains", TCS 79, (359-363).

Kleene S. C. [1945] "On the interpretation of intuitionistic number theory", JSL, 10, (109-124).

Kleene S. C. [1952] "Introduction to Metamathematics", reprinted by North Holland.

Krivine [1991] "Lambda-calcul, types et modèles", Masson.

Lambek J. [1980] "From λ-calculus to cartesian closed categories", in To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Seldin&Hindley (eds.), (403-450), Academic Press.

Lambek J., Scott P.J. [1986] "Introduction to higher order Categorial Logic", Cambridge University Press.

Longo G., Moggi E. [1984] "Cartesian Closed Categories of Enumerations and effective Type Structures", Symposium on Semantics of Data Types (Kahn, MacQueen, Plotkin eds.), LNCS 173, Springer-Verlag, (235-247).

Longo G., Moggi E. [1992] "Constructive Natural Deduction and its Modest Interpretation", in

MSCS.

Mac Lane S. [1971] "Categories for the working mathematician", Springer-Verlag, New York.

Markowsky G. [1976] "Chain-complete p.o. sets and directed sets with applications", Algebra Universalis 6.

Martin-Löf P. [1984] "Intuitionistic Type Theory", Bibliopolis, Napoli.

Meyer A. R. [1982] "What is a model of the lambda calculus?", Information and Control 52, (87-122).

Milner R. [1977] "Fully abstract models of typed lambda-calculi", TCS,4(1), (1-23).

Milner R. [1989] "Communication and Concurrency", Prentice-Hall.

Mitchell J., Moggi E. [1991] "Kripke-style models for typed lambda calculus", APAL 51, (99-124), ext. abs. in LICS 87.

Moggi E. [1988] "Partial morphisms in categories of effective objects", Info.&Comp., 76, (250-277).

Moggi E. [1989] "Computational lambda-calculus and monads", in Proc. 4th IEEE-LICS.

Mulmuley K. [1987] "Full abstraction and semantic equivalence", MIT Press.

Murthy C. [1991] "An evaluation semantics for classical proofs", in Proc. IEEE-LICS, Amsterdam.

Myhill J., Shepherdson J. [1955] "Effective operations on partial recursive functions", Zeit. für Math. Logik und Grund. der Math., 1, (310-317).

O'Hearn P., Tennent R. [1993] "Relation parametricity and local variables", in Proc. ACM-POPL 93.

Palmgren E., Stoltenberg V. [1990] "Domain interpretations of Martin-Löf partial type theory", APAL, 48, (135-196).

Paulson L.C. [1987] "Logic and Computation, Interactive proof with Cambridge LCF", Cambridge University Press.

Peyton-Jones S. [1987] "The implementation of functional programming languages", Prentice-Hall.

Phoa W. [1990] "Effective domains and intrinsic structure", 5th IEEE-LICS, Philadelphia.

Plotkin G. [1975] "Call-by-name, call-by-value and the lambda-calculus", Theor. Computer Sci., 1, (125-159).

Plotkin G. [1976] "A powerdomain construction", SIAM J. of Computing, 5, 3, (452-487).

Plotkin G. [1977] "LCF as a programming language", TCS 5, (223-257).

Plotkin G. [1983] "Domains", lecture notes, C.S. Dept., Edinburgh.

Plotkin G. [1985] "Denotational semantics with partial functions", lecture notes, CSLI, Stanford 1985.

Prawitz D. [1965] "Natural deduction", Almqvist and Wiksell, Stockholm.

Reynolds J. [1974] "Towards a theory of type structures", Colloque sur la Programmation, LNCS 19, Springer-Verlag, (408-425).

Reynolds J. [1984] "Polymorphism is not set-theoretic", Symposium on Semantics of Data Types, (Kahn, MacQueen, Plotkin, eds.) LNCS 173, Springer-Verlag.

Robinson E., Rosolini P. [1988] "Categories of partial maps", Info&Co., 79, (95-130).

Rogers H. [1967] "Theory of recursive functions and effective computability", MacGraw Hill.

Rothe M. [1991] "Retraktionen auf stetigen Bereichen", Diplomarbeit am Fachbereich Mathematik der Technischen Hochschule Darmstadt, April 1991.

Rosolini G. [1986] "Continuity and effectiveness in Topoi", PhD Thesis, Oxford University.

Scott D. [1972] "Continuous lattices", Toposes, Algebraic Geometry and Logic, (Lawvere ed.), SLNM 274, (97-136).

Scott D. [1976] "Data types as lattices", SIAM Journal of Computing, 5, (522-587).

Scott D. [1980] "Relating theories of the lambda-calculus", in To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Seldin&Hindley (eds.), (403-450), Academic Press.

Scott D. [1980b] "A space of retracts", Manuscript, Bremen.

Seely R.A.G. [1984] "Locally cartesian closed Categories and type theory" Math. proc. Cambridge Phil. Soc., 95, 33, (33-48).

Seely R.A.G. [1987] "Categorical semantics for higher order polymorphic lambda calculus", JSL, 52, 4, (969-989).

Soare R. [1987] "Recursively enumerable sets and degrees", Springer-Verlag.

Smyth M. [1977] "Effectively Given Domains", Theoret. Comput. Sci. 5, (255-272).

Smyth M. [1978] "Power domains", JCSS, 16, (23-36).

Smyth M. [1983] "The largest cartesian closed category of domains", TCS, 27, (109-119).

Smyth M., Plotkin G. [1982] "The category-theoretic solution of recursive domain equations", SIAM Journal of Computing, 11, (761-783).

Stoy J. [1977] "Denotational Semantics", M.I.T. Press.

Taylor P. [1990] "An algebraic approach to stable domains", J. of Pure and Apl. Algebra, 64, (171-203).

Troelstra A. [1973] "Metamathematical investigation of Intuitionistic Arithmetic and Analysis", LNM 344, Springer-Verlag, Berlin.

Troelstra A.S. and van Dalen D. (eds) [1982] "The L.E.J. Brouwer Centenary Simposium," Studies in Logic and the Foundations of Mathematics, vol. 110, North-Holland, Amsterdam.

Troelstra A.S., van Dalen D. [1988] "Constructivism in mathematics", North Holland (2 vols.).

Vickers S. [1989] "Topology via logic", Cambridge University Press.

Wadsworth C.P. [1976] "The relation between computational and denotational properties for Scott's D∞-models of the lambda-calculus", S.I.A.M. J. Computing, 5, (488-521).

Wand M. [1979] "Fixed-point Constructions in Order-enriched Categories", Theoret. Comp. Sci. 8, (13-30).

Winskel G. [1986] "Event structures", in SLNCS 255.

Zhang G. [1991] "Logic of domains", Birkhäuser.

Appendix 1: Memento of Recursivity

Notations: Function means always partial, unless otherwise specified. $\downarrow$ is used for "is defined". A definition of the form "$f(x)\downarrow y$ iff P" has to be read " $f(x)\downarrow$ iff P, and P implies $f(x)=y$". We abbreviate $x_1,\ldots,x_n$ into $\mathbf{x}$. We also write, for two expressions s and t,

$$s\cong t \Leftrightarrow_\Delta (s\downarrow,\ t\downarrow \text{ and } s=t) \text{ or } (s\uparrow \text{ and } t\uparrow)\ .$$

Partial recursive, or computable functions, may be defined in a number of equivalent ways. This is what Church's thesis is about: all definitions of computability turn out to be equivalent. Church's thesis gives confidence in "semi-formal" arguments, used to show that a given function is computable. These arguments can be accepted only if at any moment, upon request, the author of the argument is able to fully formalize it in one of the available axiomatizations.

The most basic way of defining computable functions is by means of a computing device. Turing machines are the most well known. In Cutland[80] the reader will find a somewhat handier formalism: Unlimited Register Machines, which are simple imperative programs. Either of these formalisms leads to a notion of *computable* functions from ω^n to ω, for each n. More precisely, a Turing machine defines, for each n, a partial function $f:\omega^n\to\omega$.

More mathematical presentations are by means of recursive program schemes, or by means of combinations from basic recursive functions, like in the following

Theorem *(Gödel-Kleene)*

For any n, the set of Turing computable functions from ω^n to ω is the set of *partial recursive functions* from ω^n to ω, where by definition the class of partial recursive (p.r.) functions is the smallest class containing:

- $\mathbf{0}$: $\omega\to\omega$ defined by: $\mathbf{0}(x)=0$
- **succ** : $\omega\to\omega$ (the successor function)
- projections $\pi_i^n : \omega^n\to\omega$ defined by: $\pi_i^n(x_1,\ldots, x_n)=x_i$

and closed under the following constructions

- composition: If $f_1:\omega^m\to\omega, \ldots, f_n:\omega^m\to\omega$ and $g:\omega^n\to\omega$ are p.r. , then $g[f_1,\ldots,f_n]$ is p.r., where $g[f_1,\ldots,f_n](\mathbf{x}) = g(f_1(\mathbf{x}),\ldots,f_n(\mathbf{x}))$ (when n=1, we write $g\circ f_1$ in place of $g[f_1]$)
- primitive recursion: if $f:\omega^n\to\omega$, $g:\omega^{n+2}\to\omega$ are p.r. , then so is h defined by
  $h(\mathbf{x},0)=f(\mathbf{x})$
  $h(\mathbf{x},y+1)=g(\mathbf{x},y,h(\mathbf{x},y))$
- minimalization: if $f:\omega^{n+1}\to\omega$ is p.r. , so is g defined by
  $g(\mathbf{x})=\mu y\ (f(\mathbf{x},y)=0)$, where μy P means: the smallest y s.t. P. □

Terminology: The source of partiality lies in minimalization. The total functions

obtained by the combinations of *Gödel-Kleene*, except minimalization, are called *primitive recursive*. The partial recursive functions which are total are called the *recursive* functions. The set of partial recursive functions from ω^n to ω is called PR^n (we write PR for PR^1).

Fact: Consider the three following functions:
- $<,>:\omega\times\omega\to\omega$ defined by: $<m,n> = 2^m(2n+1) - 1$
- $\pi_1:\omega\to\omega$ where $\pi_1(n)$ is the exponent of 2 in the prime decomposition of n+1
- $\pi_2:\omega\to\omega$ defined by: $\pi_2(n) = ((n+1)/2^{\pi_1(n)} - 1)/2$

They are recursive and provide inverse bijections between $\omega\times\omega$ and ω. A function $f:\omega\times\omega\to\omega$ is p.r. iff $f[\pi_1,\pi_2]:\omega\to\omega$ is p.r. . □

Similarly, one can encode Turing machines as natural numbers. Call
- T_n the Turing machine which has code n
- ϕ_n^m the partial function from ω^m to ω defined by T_n (we write ϕ_n for ϕ_n^1)
- $W_n^m \triangleq dom(\phi_n^m)$ (we write W_n for W_n^1)

If $f=\phi_n^m$ ($W=W_n^m$), we say that n is an index of f (W). Thus we have

Fact *(Enumeration of* PR*)*
The mapping $\lambda n.\ \phi_n$ is a surjection of ω onto PR.

As a first consequence, there are total functions which are not recursive.

Exercise TOT¬REC: Show that f defined by
- $f(n) = \phi_n(n)+1$ if $\phi_n(n)\downarrow$, 0 otherwise

is not recursive. (But g defined by $g(n)\downarrow\phi_n(n)+1$ iff $\phi_n(n)\downarrow$ is p.r, see following theorem *Universal*). Show that there exist recursive, non primitive recursive functions.

The next theorem says that arguments of a partial recursive function can be frozen, uniformly.

Theorem *(s-m-n)*:
For each m,n there is a total recursive m+1 ary function s_n^m (s for short) s.t. for all $\mathbf{x}=x_1,\dots,x_m$, $\mathbf{y}=y_{m+1},\dots,y_{m+n}$ and p: $\phi_p^{m+n}(\mathbf{x},\mathbf{y}) \cong \phi_{s(p,\mathbf{x})}^n(\mathbf{y})$
Proof
Hint: we can "prefix" to T_p instructions that input the frozen argument $\mathbf{x}$. □

Theorem *(Universal)*
There exists a Turing machine T_U computing, for any n, the function $\psi_U^n:\omega^n\to\omega$ defined by: $\psi_U^n(p,\mathbf{y}) \triangleq \phi_p^n(\mathbf{y})$.

Proof

Hint: Informally, T_U decodes its first argument p into the machine T_p, and then acts as T_p on the remaining arguments. □

Computability specializes to predicates:

Definition *(Decidable and semi-decidable)*

A subset W of ω^n is called *decidable*, or *recursive*, when its characteristic function χ ($\chi(x)=0$ if $x\in W$, 1 otherwise) is recursive.

A subset W of ω^n is called *primitive recursive*, when its characteristic function is primitive recursive.

A subset W of ω^n is called *semi-decidable*, or *recursively enumerable* (r.e.), when its partial characteristic function χ_p ($\chi_p(x)\downarrow 1$ iff $x\in W$) is partial recursive.

A central example of a recursive set is:

Fact (*Convergence in t steps*)

Given a partial recursive function f, $\{(x,y,t) \mid f(x)\downarrow y$ in t steps$\}$ is recursive.

Proof

Given a Turing machine T computing f, the obvious informal algorithm is: perform t steps of T starting with input x, and check whether result y has been reached. □

Remark: A more careful analysis shows that the characteristic function of $\{(x,y,t) \mid f(x)\downarrow y$ in t steps$\}$ can be defined by means of primitive recursion only.

There are a number of equivalent characterizations of recursive and recursively enumerable sets.

Proposition *(Various definitions of r.e.)*

$W\subseteq\omega^n$ is r.e. iff one of the following conditions hold:

(1) W=dom(f), for some partial recursive function f.

(2) There exists a recursive set $W'\subseteq\omega^{n+1}$ s.t. $W=\{x \mid \exists y\ (x,y)\in W'\}$

(3) $W=\varnothing$ or W=im(h), for some recursive function $h:\omega^n\to\omega$

(4) W=im(h), for some partial recursive function $h:\omega^n\to\omega$.

Proof

(1) If W=dom(f), then its partial characteristic function is $\mathbf{1}\circ f$, where $\mathbf{1}$ is constant 1.

(2) Let W be $\{x \mid \exists y\ (x,y)\in W'\}$. Then $W=\mathrm{dom}(\lambda x.\mu y\ (x,y)\in W')$. Conversely, if W=dom(f), take $W'=\{(x,y) \mid f(x)\downarrow$ in y steps$\}$.

(3) If $W=\mathrm{dom}(f)\neq\varnothing$, pick an element $a\in W$; define $g(x,y)=x$ if $f(x)\downarrow$ in y steps, a otherwise: then W=im(g)=im(h) (where h is the composition of g with the

encoding from ω^n to ω^{n+1}).

(4) If W=im(h), we have by fact *Convergence in t steps* that {(x,y,t) | h(x)↓y in t steps} is recursive. Thus W=dom(**λx**.μz. (z=<y,t> and h(x)↓y in t steps)). □

Exercise RE∃: Show that if $W\subseteq\omega^{n+1}$ is r.e. , then {x | ∃y (x,y)∈W} is r.e.

Proposition *(Various definitions of decidability)*

$W\subseteq\omega^n$ is recursive iff one of the following conditions hold:

(1) W=dom(f), for some recursive function.

(2) W and its complement ¬W are decidable.

Proof

(1) Same argument as for proposition *Various definitions of r.e.* (1).

(2) If W is decidable, it is semidecidable, and ¬W is decidable (with characteristic function $\neg\circ\chi$, where χ is the characteristic function of W). Conversely, if W and ¬W are both semi-decidable, let W' and W'' be recursive and s.t.

W={x | ∃y (x,y)∈W'} and ¬W={x | ∃y (x,y)∈W''}.

Let χ' and χ'' be the characteristic functions of W' and W'', respectively. Then

W=dom(**λx**.μy. (**Y**[χ',χ''](x,y)=0))

where **Y** is any recursive function restricting to the boolean union over {0,1}. **λx**.μy. ($\cup$[χ',χ''](x,y)=0) is p.r. by construction, and moreover is total since W∪¬W=ω^n. □

The following is a useful characterization of partial recursive functions

Proposition *(p.r. ⇔ graph if r.e.)*

A function f is p.r. iff its *graph* {(x,y) | f(x)↓y} is r.e.

Proof

If f is p.r. , then by fact *Convergence in t steps*, {(x,y,t) | f(x)↓y in t steps} is recursive. We conclude by proposition *Various definitions of r.e.* (2) that {(x,y) | f(x)↓y} is r.e. , since f(x)↓y iff f(x)↓y in t steps for some t.

Conversely, if {(x,y) | f(x)↓y} is r.e. , let W' be a recursive set s.t. f(x)↓y iff (x,y,t)∈W' for some t. Then f can be written as $\boldsymbol{\pi_1}\circ$(**λx**.μz. (z=<y,t> and (x,y,t)∈W')), and thus is p.r. . □

Remark: The encodings quoted among others in the proofs of (3) in proposition *Various definitions of semi-decidable* "hide" a useful technique, known as "dovetailing": the informal way of obtaining h is by trying the first step of f(1), the first step of f(2), the second step of f(1), the first step of f(3), the second step of f(2), the third step of f(1), the first step of f(4)...

Here is an example of a semi-decidable, non decidable predicate:

Fact *(A non r.e. subset)*
(1) The set $K \triangleq \{x \mid x \in W_x\}$ is semi-decidable.
(2) The set $\{x \mid x \notin W_x\}$ is not r.e. .
Proof
(1) We have $K=im(\lambda x.\phi_x(x))=im(\psi_U[id,id])$, thus K is r.e. by proposition *Various definitions of r.e.* (4).
(2) Suppose: $\{x \mid x \notin W_x\}=dom(f)$ for some PR function. Let n be an index of f. We have: $\forall x,\ x \notin W_x$ iff $x \in W_n$. We get a contradiction when taking x=n. □

Exercise {REC}NOT RE : Show that $\{x \mid \phi_x$ is recursive$\}$ is not r.e.

We end up this memento with an important theorem, widely used in theoretical computer science. It gives evidence to the thesis: "computable implies continuous".

A partial function θ s.t. dom(θ) is finite is called finite. Clearly finite functions from ω to ω are computable. Partial functions may be ordered as follows:

$$f \leq g \Leftrightarrow_\Delta \forall x\ (f(x)\downarrow y) \Rightarrow (g(x)\downarrow y)$$

Theorem *(Rice-Shapiro)*
Let **A** be a subset of PR s.t. $A=\{x \mid \phi_x \in \mathbf{A}\}$ is r.e. . Then for any partial recursive f:
$f \in \mathbf{A}$ iff there exists a finite function $\theta \leq f$ s.t. $\theta \in \mathbf{A}$.
Proof
Let T be a machine computing the partial characteristic function of $K=\{x \mid x \in W_x\}$.
(⇐) Suppose: $f \in \mathbf{A}$, and $(\forall \theta \leq f\ \theta \notin \mathbf{A})$. Let g be the partial recursive function defined by
- $g(z,t)\downarrow y$ iff T starting with z does not terminate in less than t steps, and $f(t)\downarrow y$

One has, by definition of g:
- $\lambda t.g(z,t)=f$ if $z \notin K$
- $\lambda t.g(z,t)=\theta$, for some finite $\theta \leq f$, if $z \in K$

Thus our assumption entails $z \notin K \Leftrightarrow \lambda t.g(z,t) \in \mathbf{A}$. Let s be a recursive function, given by the **s-m-n** theorem, s.t. $g(z,t) \equiv \phi_{s(z)}(t)$. The above equivalence can be rephrased as:
$z \notin K \Leftrightarrow s(z) \in A$.
But the predicate on the right is r.e.: contradiction.
(⇒) Suppose: $f \notin \mathbf{A}$ and $\theta \in \mathbf{A}$, for some finite $\theta \leq f$. We argue as in the proof of (⇐), defining now g by: $g(z,t)\downarrow y$ iff $(\theta(t)\downarrow$ or $z \in K)$ and $f(t)\downarrow y$. □

Remark $\perp\mathbf{A}$: Let **A** be as in the statement of Rice-Shapiro theorem, and let ⊥ be the totally undefined function. If $\perp \in \mathbf{A}$, then, by the theorem, **A** must be the whole of PR.

Corollary *(Rice)*

If $\mathbf{B} \subseteq PR$, $\mathbf{B} \neq \emptyset$ and $\mathbf{B} \neq PR$, then $\{x \mid \phi_x \in \mathbf{B}\}$ is undecidable.

Proof

Suppose that $\{x \mid \phi_x \in \mathbf{B}\}$ is decidable. Then $\mathbf{B}$ and $\neg\mathbf{B}$ both satisfy the conditions of the Rice-Shapiro theorem. Consider the totally undefined function $\perp$. We have: $\perp \in \mathbf{B}$ or $\perp \in \neg\mathbf{B}$. By the remark $\perp\mathbf{A}$, we deduce that either $\mathbf{B}=PR$ or $\neg\mathbf{B}=PR$: contradiction. □

Hints for exercises:

TOT$\neg$REC (last part). Use an enumeration $(\theta_n)_{n\in\omega}$ of the primitive recursive functions, and take $\lambda x.\ \theta_x(x)+1$.

RE$\exists$: Consider a recursive W' s.t. $W=\{(\mathbf{x},y) \mid \exists z\ (\mathbf{x},y,z) \in W'\}$ is r.e. .

{REC}NOT RE: consider $g(x)=\phi_{f(x)}(x)+1$, where f is a claimed enumeration.

Appendix 2: Basic Category Theory

Contents: 1. Basic Definitions, 2. Limits, 3. Functors and Natural Transformations, 4. Universal Arrow and Adjunction, 5. Equivalence and Reflection, 6. Adjoints and Limits, 7. Cartesian Closed Categories, 8. Monads.

Category theory has been tightly connected to abstract mathematics since the first paper on cohomology by Eilenberg and Mac Lane (1942) which establishes its basic notions. This appendix is a pro-memoria for a few elementary definitions and results in this lively branch of mathematics. We refer to MacLane[71] and Asperti&Longo[91] for adequate introductions and wider perspectives.

In the mathematical practice category theory is helpful in: (1) Formalizing a problem, as it is a good habit to ask in which category we are working in, if a certain transformation is a functor, if a given subcategory is reflective, etcetera. (2) Formulating a result, as by using category theoretic terminology one can often express a result in a more modular and abstract way. [38]

Categorical logic is a branch of category theory that arises from the observation that logical operators can be suitably expressed by means of universal properties (Lawvere). In this way one represents the models of, say, intuitionistic propositional logic, as categories with certain closure properties where sentences are interpreted as objects and proofs as morphisms.

The tools developed in categorical logic begin to play a central role in the study of programming languages. There are at least three intermediate steps that suggest a link between these two apparently distant topics:

- The role of (typed) lambda calculi in the work of Landin, McCarthy, Strachey, and Scott on the foundation of programming languages.
- The Curry-Howard correspondence between systems of natural deduction and typed lambda calculi.
- The categorical semantics of typed lambda calculi along the lines traced by Lambek and Scott.

The basic idea is to represent the "models" of programming languages as certain abstract structures described in a category theoretic language. The practical fall-outs that we expect from such a "model theory" are *effective and complete* theories to reason about programs (see, e.g., Moggi[91]).

This approach has been fairly successful in describing data types by means of universal properties. At present it is unclear if this program will be successful on a larger variety of programming languages features. It is however a recognized fact that ideas from categorical logic play a central role in the study of *functional*

[38] A list of "prescriptions" for the use of category theory in computer science can be found in Goguen[91].

languages. Moreover promising attempts to describe categorically other features of programming languages such as modules, continuations, local variables, etcetera are actively pursued. It is in this perspective that some elementary parts of category theory are introduced in this appendix.

1. Basic Definitions

A category may be regarded as a directed labelled graph endowed with a partial operation of composition of edges which is associative and has an identity.

Definition *(category)*

A category **C** is a sextuple (Ob, Mor, dom, cod, id, comp) such that:

dom: Mor→Ob, cod: Mor→Ob, id: Ob→Mor,

comp: Comp→Mor,

where: Comp = {(f, g)∈Mor×Mor | dom(f) = cod(g)}

$id \circ f = f \circ id = f$ (identity)

$f \circ (g \circ h) = (f \circ g) \circ h$ (associativity)

where we omit to write the object to which id is applied in (identity) and we write $f \circ g$ only if (f, g)∈Comp; also $f \circ g$ is a shorthand for comp(f, g).

Conventions: Let **C** be a category, a, b∈Ob, then **C**[a, b] = {f∈Mor | dom(f) = a ∧ cod(f) = b}. We also write f: a→b for f∈**C**[a, b], and a∈**C** for a∈Ob. When confusion may arise we decorate the components Ob, Mor, ... of a category with its name, hence writing Ob_C, Mor_C, ... A category **C** is *small* if Mor_C is a set, and it is *locally small* if for any a, b∈**C**, **C**[a, b] is a set.

Examples of categories

We just specify objects and morphisms. The operation of composition is naturally defined. The verification of the identity and associativity laws is immediate: (1) Sets and Functions. (2) Sets and Partial Functions. (3) Sets and Binary Relations. (4) Every pre-order (P, ≤) induces a category **P** with #**P**[a, b] =1 if a≤b and **P**[a, b] =∅ otherwise. (5) One object and morphisms as elements of a monoid. (6) Pre-Orders (or Posets) and Monotone Functions. (7) Groups and Homomorphisms. (8) Topological Spaces and Continuous Functions. (9) Any set with just an identity morphism for each object (this is the "discrete" category). (10) Oriented Graphs and Transformations that preserve domain and codomain of edges.

Definition *(dual category)*

Let **C** be a category. We define a dual category $\mathbf{C}^{op}$ as follows:

$Ob_{C^{op}} = Ob_C$ $\quad \mathbf{C}^{op}[a, b] = \mathbf{C}[b, a]$

$id^{op} = id$ $\quad f \circ^{op} g = g \circ f$.

Note *(dual property)*

Given a property P for a category **C** and relative theorems it often makes sense to consider a "dual property" P^{op} to which correspond "dual theorems". This idea can be formalized using the notion of dual category as follows: given a property P for a category **C** we say that **C** has property P^{op} if $\mathbf{C}^{op}$ has property P.

Examples of categories built out of categories

(1) Subcategory: any sub-graph of a given category closed by composition and identity. (2) If **C** and **D** are categories the product category $\mathbf{C}\times\mathbf{D}$ is defined as: $Ob_{\mathbf{C}\times\mathbf{D}} = Ob_{\mathbf{C}}\times Ob_{\mathbf{D}}$, $\mathbf{C}\times\mathbf{D}[(a,b), (a',b')] = \mathbf{C}[a, a']\times\mathbf{D}[b, b']$. (3) If **C** is a category and $a\in\mathbf{C}$, the slice category $\mathbf{C}\downarrow a$ is defined as: $\mathbf{C}\downarrow a = \cup_{b\in\mathbf{C}}\mathbf{C}[b, a]$, $\mathbf{C}\downarrow a[f, g] = \{h \mid g\circ h = f\}$.

Definition *(properties of objects and morphisms)*

Let **C** be a category.

An object a is *terminal* if $\forall b.\ \exists! f: b\to a$. We denote a terminal object with 1 and with $!_b$ the unique morphism from b to 1.

A morphism $f: a\to b$ is a *mono* if $\forall h, k.\ f\circ h = f\circ k \Rightarrow h=k$.

A morphism f is *epi* if it is mono in $\mathbf{C}^{op}$, i.e. $\forall h, k.\ h\circ f = k\circ f \Rightarrow h=k$.

A mono $f: a\to b$ is *split* if there is $g: b\to a$ such that $g\circ f = id$.

A morphism $f: a\to b$ is an *iso* if there is $g: b\to a$ such that $g\circ f = id$ and $f\circ g = id$. We write $a\cong b$ if there is an iso between a and b.

Exercises: Prove the following properties: (1) each object has a unique identity morphism, (2) the inverse of an iso is unique, (3) if $g\circ f = id$ then g is an epi and f is a mono, (4) f split mono and epi implies f iso, (5) f mono and epi does not imply f iso. (6) The terminal object is uniquely determined up to isomorphism.

2. Limits

The notions of cone and limit of a diagram are presented. The main result explains how to build limits of arbitrary diagrams by combining limits of special diagrams, namely products and equalizers.

Definition *(diagram)*

Let **C** be a category and $I = (Ob_I, Mor_I)$ a graph. A diagram in **C** over I is a graph morphism $D: I\to\mathbf{C}$.

Convention: We often represent a diagram D as a pair $(\{d_i\}_{i\in Ob_I}, \{f_u\}_{u\in Mor_I})$.

Definition *(category of cones)*

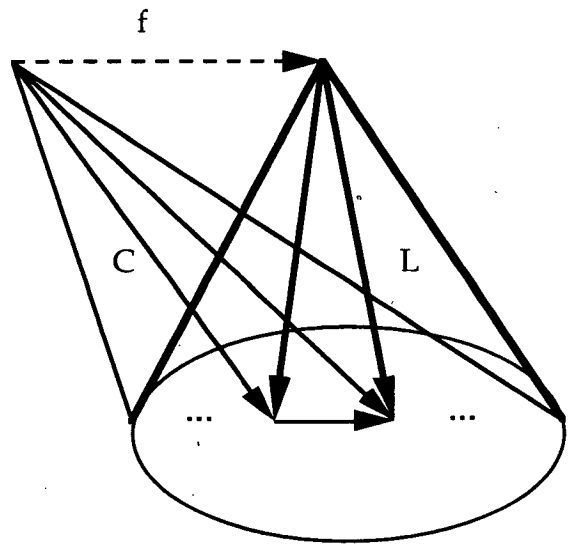
Let **C** be a category and D: I→**C** a diagram. We define the category of cones $\mathrm{Cones}_{\mathbf{C}}\mathrm{D}$ as follows:

$$\mathrm{Cones}_{\mathbf{C}}\mathrm{D} = \{(c, \{h_i\}_{i\in \mathrm{Ob}_I}) \mid \forall u\in \mathrm{Mor}_I.\ f_u: d_i\to d_j \Rightarrow h_j = f_u\circ h_i\}$$

$$\mathrm{Cones}_{\mathbf{C}}\mathrm{D}[(c, \{h_i\}_{i\in \mathrm{Ob}_I}), (d, \{k_i\}_{i\in \mathrm{Ob}_I})] = \{g: c\to d \mid \forall i\in \mathrm{Ob}_I.\ h_i = k_i\circ g\}.$$

Definition *(limit)*

Let **C** be a category and D: I→**C** a diagram. D has a limit if the category $\mathrm{Cones}_{\mathbf{C}}\mathrm{D}$ has a terminal object. The following diagram illustrates this definition.



Note By the properties of terminal objects it follows that limits are determined up to isomorphism in $\mathrm{Cones}_{\mathbf{C}}\mathrm{D}$. Hence we may improperly speak of a limit as an object of the category $\mathrm{Cones}_{\mathbf{C}}\mathrm{D}$, we denote this object by $\lim_{\mathbf{C}}\mathrm{D}$. Also we say that the category **C** has I-limits if all the diagrams indexed over I have limits.

Note *(dual notions)*

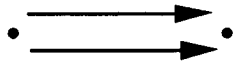
The notions of cocone, initial object, and colimit are dual to the notions of cone, terminal object, and limit, respectively.

Examples *(limits)*

(1) If $I = \varnothing$ then the limit is a *terminal object.*

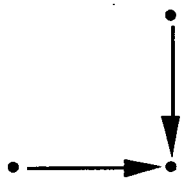
(2) If I is a discrete graph (no morphisms) then a diagram over I in **C** is just a family of objects $\{a_i\}_{i\in \mathrm{Ob}_I}$. In this case a limit is also called a *product* and it is determined by a couple $(c, \{\pi_i: c\to a_i\}_{i\in \mathrm{Ob}_I})$ such that for any cone $(d, \{f_i: c\to a_i\}_{i\in \mathrm{Ob}_I})$ there exists a unique z: d→c such that $\forall i\in \mathrm{Ob}_I.\ f_i = \pi_i\circ z$. Traditionally one can write c as $\prod_{i\in \mathrm{Ob}_I} a_i$, and z as $\langle f_i\rangle$.

(3) *Equalizers* are limits of diagrams over a graph I with the following structure:



If the image of the diagram is a pair of maps f, g: a→b then an equalizers (or limit) is a pair (c, e: c→a) with properties (i) f∘e = g∘e, and (ii) ((c', e': c'→a) and f∘e' = g∘e') implies ∃!z: c'→c. (e∘z = e').

(4) *Pullbacks* are limits of diagrams over a graph I with the following structure:



If the image of the diagram is a pair of maps f: a→d, g: b→d then a pullback (or limit) is a pair (c, {h: c→a, k: c→b}) with properties (i) f∘h = g∘k, and (ii) ((c', {h': c→a, k': c→b}) and f∘h' = g∘k') implies ∃!z: c'→c. (h∘z = h' ∧ k∘z = k').

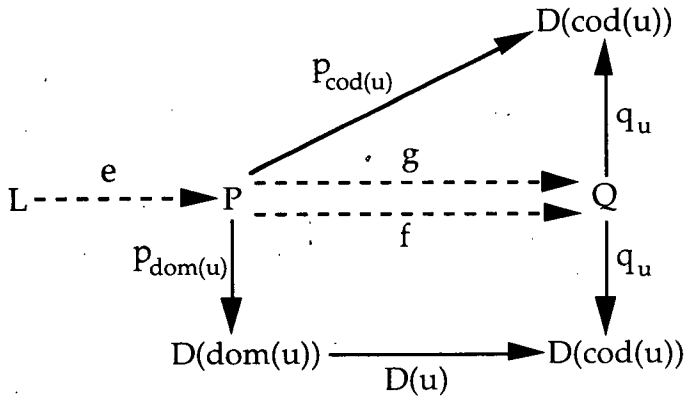
Exercise: Show that a category with terminal object and pullbacks has binary products and equalizers.

Theorem *(sufficient condition for the existence of I-limits)*

Let **C** be a category and I a graph, then **C** has I-limits if (1) **C** has equalizers, (2) **C** has all products indexed over Ob_I or Mor_I. In particular a category with equalizers and finite products has all finite limits.

Proof

Let D: I→**C** be a diagram. We define $P = \prod_{i \in Ob_I} d_i$, and $Q = \prod_{u \in Mor_I} cod(D(u))$. Next we define f, g, and e according to the following diagram:

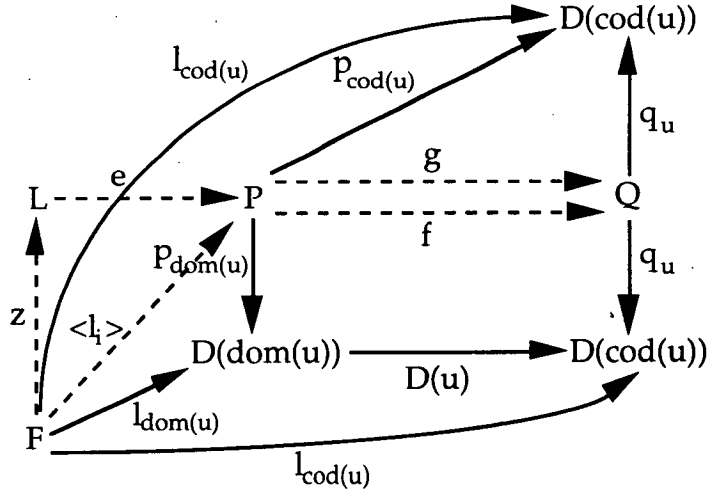


where: (i) p and q denote the projections of P and Q respectively. (ii) f is the unique map such that $D(u) \circ p_{dom(u)} = q_u \circ f$, for any $u \in Mor_I$. (iii) g is the unique map such that $p_{cod(u)} = q_u \circ g$, for any $u \in Mor_I$. (iv) e is the equalizer of f and g.

We claim that $(L, \{p_i \circ e\}_{i \in Ob_I})$ is a limit of the diagram D. The proof of this fact takes several steps.

(1) $(L, \{p_i \circ e\}_{i \in Ob_I}) \in Cones_C D$. We have to show: $D(u) \circ p_{dom(u)} \circ e = p_{cod(u)} \circ e$, for any $u \in Mor_I$. Observe: $D(u) \circ p_{dom(u)} \circ e = q_u \circ f \circ e = q_u \circ g \circ e = p_{cod(u)} \circ e$.

(2) Let $(F, \{l_i\}_{i\in Ob_I}) \in Cones_C D$. Then there is a uniquely determined map $\langle l_i\rangle$: $F \to P$ such that: $p_i \circ \langle l_i\rangle = l_i$, for any $i \in Ob_I$.
(3) We claim $f \circ \langle l_i\rangle = g \circ \langle l_i\rangle$. This follows by the observation that for any $u \in Mor_I$:
$q_u \circ f \circ \langle l_i\rangle = D(u) \circ p_{dom(u)} \circ \langle l_i\rangle = D(u) \circ l_{dom(u)} = l_{cod(u)} = p_{cod(u)} \circ \langle l_i\rangle = q_u \circ g \circ \langle l_i\rangle$.
(4) Hence there is a unique $z: F \to L$ such that $e \circ z = \langle l_i\rangle$.
(5) We verify that $z: (F, \{l_i\}_{i\in Ob_I}) \to (L, \{p_i \circ e\}_{i\in Ob_I})$ in $Cones_C D$ by checking: $p_i \circ e \circ z = l_i$, for any $i \in Ob_I$. This follows by: $p_i \circ e \circ z = p_i \circ \langle l_i\rangle = l_i$.
(6) Finally suppose z': $(F, \{l_i\}_{i\in Ob_I}) \to (L, \{p_i \circ e\}_{i\in Ob_I})$ in $Cones_C D$. Then
z': $(F, \langle l_i\rangle) \to (L, e)$ as: $p_i \circ e \circ z' = l_i$, for any $i \in Ob_I$ implies $e \circ z' = \langle l_i\rangle$. Hence $z = z'$.
The following diagram represents the constructions carried on:



□

Exercise: Study the existence of (co-)limits in the categories introduced in section 1.

3. Functors and Natural Transformations

A functor is a morphism between categories and a natural transformation is a morphism between functors. The main result presented here is that there is a full and faithful functor from any category **C** to the category of set-valued functors over $\mathbf{C}^{op}$.

Definition *(functor)*

Let **C**, **D** be categories, a functor $F: \mathbf{C} \to \mathbf{D}$ is a morphism between the underlying graphs that preserves identity and composition, that is:

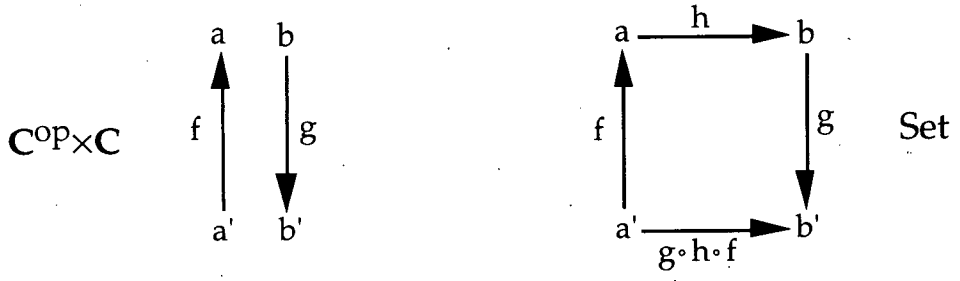
$F_{Ob}: Ob_C \to Ob_D$, $\quad F_{Mor}: Mor_C \to Mor_D$,
$f: a \to b \Rightarrow F_{Mor}(f): F_{Ob}(a) \to F_{Ob}(b)$,
$F_{Mor}(id_a) = id_{F_{Mor}(a)}$ $\quad F_{Mor}(f \circ g) = F_{Mor}(f) \circ F_{Mor}(g)$.

Conventions: In the following we omit the indices "Ob" and "Mor" in a functor. By a *contravariant functor* $F: \mathbf{C} \to \mathbf{D}$ we mean a functor $F: \mathbf{C}^{op} \to \mathbf{D}$.

Exercise: Show that small categories and functors form a category.

Definition *(hom-functor)*

Let **C** be a locally small category. We define the hom-functor $C[\ ,\ ]: \mathbf{C}^{op}\times\mathbf{C}\to Set$ as follows: $C[\ ,\ ](a, b) = C[a, b]$ $C[\ ,\ ](f, g) = \lambda h.\ g\circ h\circ k$



Convention: Given an object c in the category **C** we denote with $C[\ ,\ c]: \mathbf{C}^{op}\to Set$, and $C[c,\ ]: \mathbf{C}\to Set$ the contravariant and covariant functors over **C** obtained by restricting the hom-functor to the first and second component, respectively.

Exercise: Suppose $F: \mathbf{C}\to\mathbf{D}$ is a functor, $D: I\to\mathbf{C}$ is a diagram, and $(a, \{l_i\}_{i\in Ob_I})\in Cones_C D$. Then $(Fa, \{Fl_i\}_{i\in Ob_I})\in Cones_D F\circ D$.

Definition *(limit-preservation)*

Suppose $F: \mathbf{C}\to\mathbf{D}$ is a functor, and $D: I\to\mathbf{C}$ is a diagram. We say that F preserves the limits of the diagram D if:

$$(a, \{l_i\}_{i\in Ob_I})\in \lim_C D \Rightarrow (Fa, \{Fl_i\}_{i\in Ob_I})\in \lim_D F\circ D.$$

Proposition *(the contravariant hom functor preserves limits)*

Let **C** be a locally small category and c an object in **C** then the covariant hom-functor $C[c,\ ]: \mathbf{C}\to Set$ preserves limits.

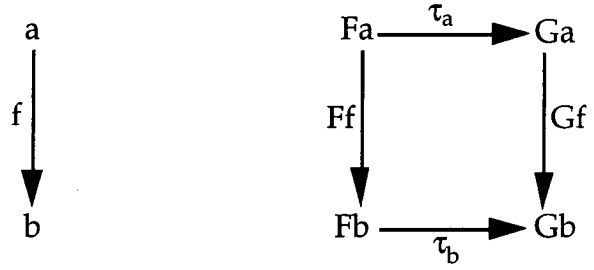
Proof

Let $D = (\{d_i\}_{i\in Ob_I}, \{f_u\}_{u\in Mor_I})$ be a diagram and $(a, \{l_i\}_{i\in Ob_I})\in \lim_C D$. Then: $(C[c, a], \{\lambda h.\ l_i\circ h\}_{i\in Ob_I})\in Cones_{Set} C[c,\ ]\circ D$. Suppose $(X, \{g_i\}_{i\in Ob_I})\in Cones_{Set} C[c,\ ]\circ D$, that is: $\forall u\in Mor_I.\ \forall x\in X.\ f_u: d_i\to d_j \Rightarrow f_u\circ g_i(x) = g_j(x)$.

Then: $\forall x\in X.\ (c, \{g_i(x)\}_{i\in Ob_I})\in Cones_C D$. Hence there is a unique $h(x): c\to a$ such that: $g_i(x) = l_i\circ h(x)$, for any $i\in Ob_I$. We can then build a unique $z: X\to C[c, a]$ such that: $l_i\circ z = g_i$, for any $i\in Ob_I$. Such z is defined by $z(x) = h(x)$. □

Definition *(natural transformation)*

Let F, G: **C**→**D** be functors. A natural transformation τ: F→G is a family $\{\tau_a: Fa \to Ga\}_{a \in Ob\mathbf{C}}$ such that for any f: a→b, $\tau_b \circ Ff = Gf \circ \tau_a$.



Exercise: Given **C**, **D** categories show that the collection of functors from **C** to **D** and natural transformations form a category. We denote this new category with $\mathbf{D}^{\mathbf{C}}$. It can be shown that $\mathbf{D}^{\mathbf{C}}$ is actually an "exponent" in the sense of cartesian closed categories (see section 7).

Definition *(category of pre-sheaves)*

Given a category **C** the category of pre-sheaves over **C** is the category $\mathbf{Set}^{\mathbf{C}^{op}}$ of contravariant set-valued functors and natural transformations.

Another important operation involving natural transformations is the composition with a functor.

Fact: If G: **B**→**C**, F, F': **C**→**C'** and δ:F→F', then δG: F∘G→F'∘G is natural, where δG is defined by set theoretic composition, i.e. $(\delta G)_a \triangleq \delta_{Ga}$. Likewise, if H: **C'**→**B'**, F, F': **C**→**C'** and δ:F→F', then Hδ: H∘F→H∘F' is natural, where $(H\delta)_a \triangleq H(\delta_a)$.

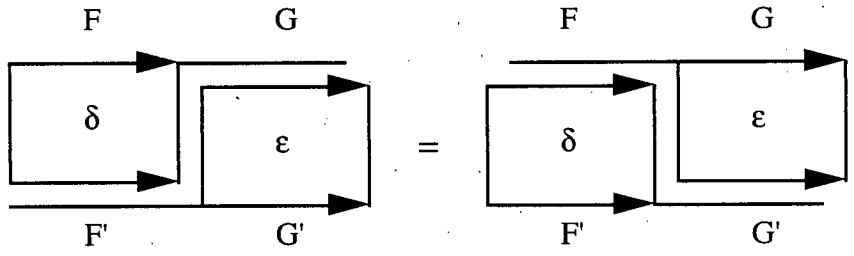
The composition of natural transformations and functors extends to a notion of "horizontal" composition of natural transformations (in contrast to the "vertical" one given by "∘").

Fact: If F, F': **C**→**C'**, G, G': **C'**→**C''**, δ:F→F', ε:G→G', then

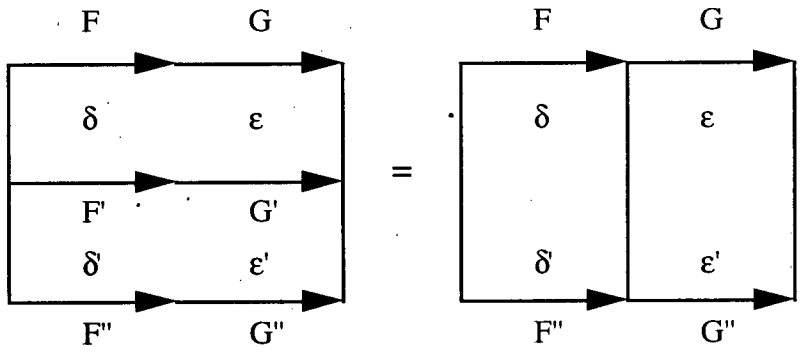
$$(\varepsilon F') \circ (G\delta) = (G'\delta) \circ (\varepsilon F): G \circ F \to G' \circ F' .$$

We write εδ for the common value of both sides of this equation.

This is illustrated in the following figure, where: (i) δ, ε are represented as rectangles whose vertical lines are irrelevant and whose horizontal lines are the source and target functors: F, F' and G, G' respectively. (ii) Composition with a functor, say Gδ, is represented by adding a line to represent G. (iii) Vertical composition is indeed represented "vertically", i.e. by matching the target of the transformation "above" with the source of the transformation "below".



Exercise: Show the following so called *interchange law* (originally stated by R. Godement), for all $\delta, \delta', \varepsilon, \varepsilon'$ of appropriate types: $(\varepsilon' \circ \varepsilon)(\delta' \circ \delta) = (\varepsilon' \delta') \circ (\varepsilon \delta)$ (as illustrated below).



Definition *(full and faithful functor)*

A functor F: $\mathbf{C} \to \mathbf{D}$ is *full* if $\forall a, b.\ \forall h{:}Fa \to Fb.\ \exists f{:}\ a \to b.\ (Ff = h)$, and it is *faithful* if it is injective on each hom-set $\mathbf{C}[a, b]$.

Theorem *(Yoneda)*

For any category **C** there is a full and faithful functor Y: $\mathbf{C} \to \mathbf{Set}^{\mathbf{C}^{op}}$ from **C** into the category of pre-sheaves $\mathbf{Set}^{\mathbf{C}^{op}}$ that is defined as follows:

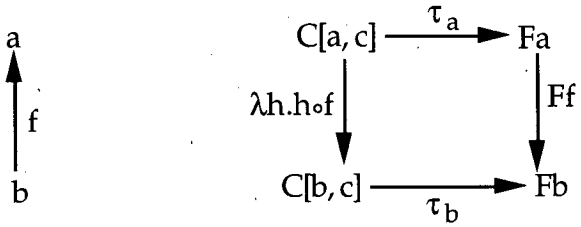
$Y(c) \triangleq \mathbf{C}[\_, c] \qquad Y(f) \triangleq \lambda h.\ f \circ h$.

Proof

The key to the proof that Y is full resides in the following lemma where we take F as h_d .

Yoneda's Lemma: For any functor F: $\mathbf{C}^{op} \to \mathbf{Set}$ and any object $c \in \mathbf{C}$ the following isomorphism holds in **Set**: $Fc \cong Nat[h_c, F]$, where $h_c \triangleq \mathbf{C}[\_, c]$, and $Nat[h_c, F]$ are the natural transformations from h_c to F.

The following diagram describes a natural transformation τ: $h_c \to F$



Define i: $Fc \to Nat[h_c, F]$ with inverse j: $Nat[h_c, F] \to Fc$ as follows:

$i(x) \triangleq \lambda d.\ \lambda h: d \to c.\ (Fh)(x)$ $\qquad$ $j(\tau) \triangleq \tau(c)(id_c)$.

- First we verify that i(x) is a natural transformation as:
  $(Ff)(i(x)(a))(h) = (Ff)(Fh)(x) = F(h \circ f)(x) = (i(x)(b))(h \circ f)$
- Next we verify that j is the inverse of i:
  $j(i(x)) = i(x)(c)(id_c) = (Fid_c)(x) = (id)(x) = x.$
  $i(j(\tau)) = \lambda d.\ \lambda h: d \to c.\ (Fh)(\tau(c)(id_c)) = \lambda d.\ \lambda h: d \to c.\ \tau(d)(h) = \tau,$

as by applying the naturality of τ to $h: d \to c$ one gets: $(Fh)(\tau(c)(id_c)) = \tau(d)(h)$. □

4. Universal Arrow and Adjunction

A universal arrow is a rather simple abstraction of a frequent mathematical phenomena.

Examples

(1) Given a signature Σ consider the category of Σ-algebras and morphisms. If A is a Σ-algebra denote with $|A|$ its carrier (which is a set). There is a well known construction which associates to any set X the "free" Σ-algebra $\Sigma(X)$ and which is characterized by the following property:

$\exists u: X \to \Sigma(X).\ \forall A.\ \forall f: X \to |A|.\ \exists! f': \Sigma(X) \to A.\ f' \circ u = f.$

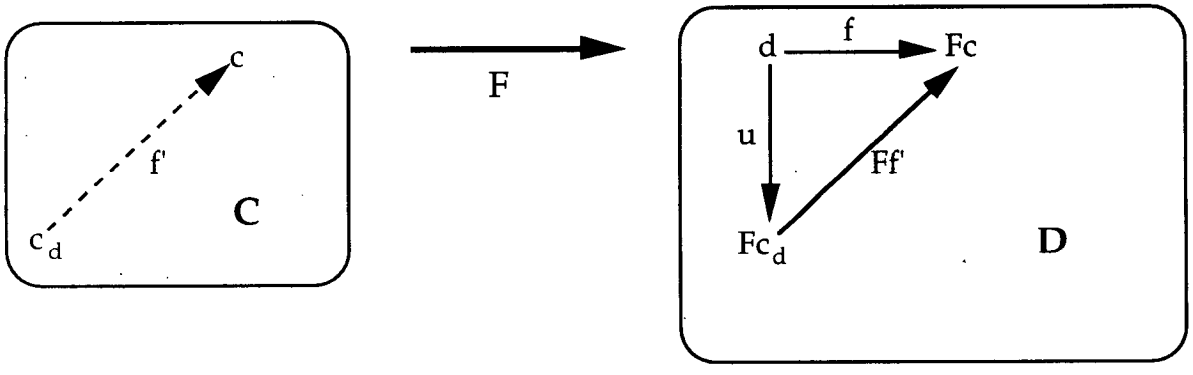
(2) Consider the category of metric spaces and continuous maps and the full subcategory of complete metric spaces. The Cauchy completion associates to any metric space (X, d) a complete metric space (X_c, d_c) which is characterized by:

$\exists u: (X, d) \to (X_c, d_c).\ \forall (Y, d')$ cmpl. $\forall f: (X,d) \to (Y,d').\ \exists! f': (X_c, d_c) \to (Y,d').\ f' \circ u = f.$

Definition *(universal arrow)*

Let $F: \mathbf{C} \to \mathbf{D}$ be a functor and d an object in D. Then the couple $(c_d, u: d \to Fc_d)$ is universal from d to F (and we also write $(c_d, u): d \to F$) if

$\forall c.\ \forall f: d \to Fc.\ \exists! f': c_d \to c.\ Ff' \circ u = f.$



Exercises (1) Show that if $(c_d, u): d \to F$ and $(c_d', u'): d \to F$ then $c_d \cong c_d'$. (2) Explicit the dual notion of co-universal. (3) Verify that the previous examples fit the definition of universal arrow.

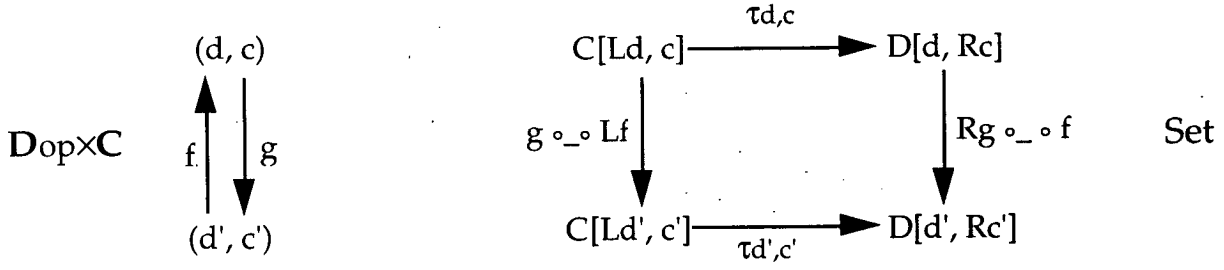
The notion of adjunction is a fundamental one, it has several equivalent characterizations. In particular it will be pointed out that an adjunction arises whenever there is a uniform way of determining a universal arrow.

Definition *(adjunction)*

An adjunction is a triple (L, R, τ), where L: **C**$\to$**D**, and R: **D**$\to$**C** are functors and τ: **C**[L_, _]$\to$**D**[_, R_] is a natural isomorphism

$$\mathbf{C} \underset{R}{\overset{L}{\rightleftarrows}} \mathbf{D}$$

We say that L is the left adjoint, R is the right adjoint, and denote this situation by L$\dashv$R. For **C**[L_, _], **D**[_, R_]: $\mathbf{D}^{op}\times\mathbf{C}\to$Set here is what the existence of a natural transformation means, where $\tau_{d,c}$, $\tau_{d',c'}$ are bijections:



Example (Poset)

In the following we develop some properties of adjunctions in the special case in which **C** and **D** are poset-categories and therefore the functors L and R are monotone functions. Let us first observe that the triple (L, R, τ) is an adjunction iff $\forall c, d.\ (Ld\leq c \Leftrightarrow d\leq Rc)$.[39]

(1) Every component of an adjunction determines the other.
Because: if L$\dashv$R and L$\dashv$R' then $d\leq Rc \Leftrightarrow Ld\leq c \Leftrightarrow d\leq R'c$. For d = Rc we get: $Rc\leq Rc \Leftrightarrow Rc\leq R'c$. Hence $Rc\leq R'c$, and symmetrically $R'c\leq Rc$. □

(2) The following conditions are equivalent for R: **C**$\to$**D**, and L: **D**$\to$**C**.
(a) $\forall c, d.\ (Ld\leq c \Leftrightarrow d\leq Rc)$.
(b) $L\circ R \leq Id_C$, $Id_D \leq R\circ L$.
(c) $L\circ R = L\circ R\circ L\circ R \leq Id_C$, $Id_D \leq R\circ L = R\circ L\circ R\circ L$.
Because: (a)$\Rightarrow$(b): $LRc \leq c \Leftrightarrow Rc \leq Rc$. (b)$\Rightarrow$(c): $L\circ R\circ L\circ R \geq L\circ Id\circ R = L\circ R$, and $L\circ R\circ L\circ R \leq Id\circ L\circ R = L\circ R$. (c)$\Rightarrow$(a): $Ld\leq c \Rightarrow d\leq RLd\leq Rc$, $d\leq Rc \Rightarrow Ld\leq LRc\leq c$. □

(3) Observe that $(c_d, d\leq Fc_d)$ is universal from d to F: **C**$\to$**D** if $\forall c'.\ (d\leq Fc' \Rightarrow c_d\leq c')$. If $\forall d.\ (c_d, d\leq Fc_d)$: d$\to$F then we can define $L(d) = c_d$, and L$\dashv$F.
Because: $d\leq Fc_d$ means $d\leq FLd$, and $Ld\leq c' \Rightarrow d \leq FLd \leq Fc'$. □

(4) Vice versa if L$\dashv$R, R: **C**$\to$**D**, and L: **D**$\to$**C** then (a) $\forall d.\ (Ld, d\leq RLd)$ is a universal from d to R, and (b) $\forall c.\ (Rc, LRc\leq c)$ is a co-universal from c to L.

[39] A pair of monotone functions satisfying this property is also known as Galois connection.

Because: it follows from (2-3). □

Exercise: The following generalizes point (1) of **Poset**. If L⊣R and L⊣R' then R and R' are naturally isomorphic.

The following theorem points out a certain amount of structure associated to any adjunction and generalizes points (2-4) of **Poset**.

Theorem

An adjunction (L, R, τ) determines:
(1) a natural transformation η: $Id_{\mathbf{D}} \to RL$ such that for each object $d \in \mathbf{D}$, (Ld, η_d) is universal from d to R, and for each f: $Ld \to c$ $\tau(f) = R(f) \circ \eta_d$: $d \to Rc$.
(2) a natural transformation ε: $LR \to Id_{\mathbf{C}}$ such that for each object $c \in \mathbf{C}$, (Rc, ε_c) is co-universal from c to L, and for each g: $d \to Rc$ $\tau^{-1}(g) = \varepsilon_c \circ L(g)$: $Ld \to c$.
(3) moreover the following diagrams commute:

$$R \xrightarrow{\eta R} RLR \xrightarrow{R\varepsilon} R \qquad \text{with } Id_R \text{ from } R \text{ to } R$$

$$L \xrightarrow{L\eta} LRL \xrightarrow{\varepsilon L} L \qquad \text{with } Id_L \text{ from } L \text{ to } L$$

Exercises: Let **C** be a category. Then
(1) **C** has a terminal object iff the unique functor !: **C**→**1** has a right adjoint.
(2) **C** has a binary products iff the diagonal functor Δ: **C**→**C**×**C** has a right adjoint.
(3) Given a graph I consider the category [I→**C**] of graphs and natural transformations (observe that the definition of natural transformation does not require I to be a category). Define a generalized diagonal functor Δ_I: **C**→[I→**C**] and show that **C** has limits of I-indexed diagrams iff the functor Δ_I has a right adjoint.
(4) Show that the left adjoint of the inclusion functor from complete metric spaces to metric spaces builds the Cauchy completion. Analogously show that the left adjoint to the forgetful functor from Σ-algebras to Sets builds the free-algebra. □

The definition of adjunction hides some redundancy, the following characterizations show different ways of optimizing it.

First Characterization. An adjunction L⊣R is completely determined by (i) a functor L: **D**→**C**, (ii) a function R: $Ob_{\mathbf{C}} \to Ob_{\mathbf{D}}$, and (iii) bijections $\tau_{d,c}$: **D**[Ld,c]→**C**[d,Rc] for all c, d, such that for all f, g, h of appropriate types: $\tau(f) \circ g = \tau(f \circ L(g))$.
(Hint: R is uniquely extended to a functor by setting $Rh = \tau(h \circ \tau\text{-}1(id))$.)

Second Characterization. An adjunction L⊣R is completely determined by (i) a functor L: **D**→**C**, (ii) a function R: $Ob_{\mathbf{C}} \to Ob_{\mathbf{D}}$, (iii) functions $\tau_{d,c}$: **D**[Ld,c]→**C**[d,Rc] for all c,d, and (iv) morphisms ε_c:L(Rc)→c (ε for short) for all c, such that for all f, g of appropriate types $\varepsilon \circ L(\tau(f)) = f$, $g = \tau(\varepsilon \circ Lg)$.

(Hint: τ is proved bijective by setting $\tau^{-1}(g)=\varepsilon\circ Lg$. The naturality is also a consequence.)

Third Characterization. An adjunction $L\dashv R$ is determined by (i) a functor L: $\mathbf{D}\to\mathbf{C}$, (ii) a function R: $Ob_C\to Ob_D$, (iii) morphisms arrows ε_c:L(Rc)$\to$c, for all c, such that for all c, d, $f\in\mathbf{C}[Ld,c]$, there exists a unique arrow, written $\tau(f)$, satisfying $\varepsilon\circ L(\tau(f))=f$.
(Hint: The naturality of the ε follows. Another way of saying this is that (Rc, ε_c) are co-universal from c to L.)

Fourth Characterization. An adjunction $L\dashv R$ is determined by (i) functors L: $\mathbf{D}\to\mathbf{C}$, R: $\mathbf{C}\to\mathbf{D}$ and (ii) natural transformations ε: LR$\to$Id, η: Id$\to$RL such that: $(\varepsilon L)\circ(L\eta)=Id_L$ and $(R\varepsilon)\circ(\eta R)=Id_R$.

5. Equivalences and Reflections

The notion of functor isomorphism is often too strong to express the idea that two categories enjoy the "same properties" (e.g. existence of limits). The following weaker notion of equivalence is more useful.

Definition *(equivalence of categories)*
A functor F: $\mathbf{C}\to\mathbf{D}$ is an equivalence of categories if there is a functor G: $\mathbf{D}\to\mathbf{C}$ such that $F\circ G\cong Id_{\mathbf{D}}$, and $G\circ F\cong Id_{\mathbf{C}}$, via natural isomorphisms.

Theorem
The following properties of a functor F: $\mathbf{C}\to\mathbf{D}$ are equivalent:
(1) F is an equivalence of categories.
(2) F is part of an adjoint (F, G, η, ε) such that η and ε are natural isomorphisms.
(3) F is full and faithful and $\forall d\in\mathbf{D}.\ \exists c\in\mathbf{C}.\ d\cong Fc$.

Exercise: Give examples of equivalent but not isomorphic pre-orders.

Reflection is a weaker condition than equivalence. The following example illustrates the idea in the poset case.

Example (Poset, continued)
(5) Suppose there is an adjunction $L\dashv R$, R: $\mathbf{C}\to\mathbf{D}$, and L: $\mathbf{D}\to\mathbf{C}$ where R is an inclusion. Then: $\forall X\subseteq\mathbf{C}\ \exists\sqcap_D X \Rightarrow \exists\sqcap_C X \wedge \sqcap_C X=\sqcap_D X$.
Because: (i) $\forall c\in X.\ (\sqcap_D X\le c) \Rightarrow \forall c\in X.\ (L(\sqcap_D X)\le Lc=c)$. (ii) $\forall c\in X.\ (y\le c)\wedge y\in\mathbf{C}\Rightarrow y\le\sqcap_D X\wedge y\in\mathbf{C}\Rightarrow y=Ly\le L(\sqcap_D X)$. □

Definition
Given Incl: $\mathbf{C}\to\mathbf{D}$ inclusion functor, we say that $\mathbf{C}$ is a reflective subcategory of $\mathbf{D}$ if there is L such that $L\dashv$Incl. L is also called the reflector functor.

The point (4) of the following theorem generalizes the previous example.

Theorem

For an adjunction (L, R, η, ε) the following holds:

(1) R is faithful iff every component ε_c: LR(c)$\rightarrow$c is an epi.

(2) R is full iff every component ε_c: LR(c)$\rightarrow$c is a split mono (i.e. it has a left inverse).

(3) Hence R is full and faithful iff ε_c: LR(c)$\rightarrow$c is an iso.

(4) If R: **C**$\rightarrow$**D** is the inclusion functor then for any diagram D: I$\rightarrow$**C** one has:

$$\exists \lim_{\mathbf{D}} D \Rightarrow \exists \lim_{\mathbf{C}} D \wedge \lim_{\mathbf{C}} D \cong \lim_{\mathbf{D}} D.$$

Examples: The full sub-category of separated topological spaces is reflective in the category of topological spaces and continuous maps. The full subcategory of posets is reflective in the category of preorders and monotone maps. On the other hand the ideal completion of a poset to a directed complete poset does not provide a left adjoint to the inclusion of directed complete posets into the category of posets and monotone maps.

6. Adjoints and Limits

Given a functor F the existence of a left (right) adjoint implies the preservation of limits (colimits). First consider the situation in **Poset**.

Example (Poset, continued)

(6) If there is an adjunction L$\dashv$R, R: **C**$\rightarrow$**D**, and L: **D**$\rightarrow$**C** then R preserves meets (and L joins).

Because: suppose X$\subseteq$**C**, and $\exists \sqcap X$. Also assume $\forall c \in X$. ($d \leq Rc$). Then: $\forall c \in X$. ($Ld \leq LRc \leq c$). Hence: $Ld \leq \sqcap X$, that implies: $d \leq RLd \leq R\sqcap X$. □

The following theorem generalizes the previous example.

Theorem

If the functor R: **C**$\rightarrow$**D** has a left adjoint then R preserves limits.

Vice versa one may wonder if the existence of limits helps in the construction of an adjunction. Consider again the situation in **Poset**.

Example (Poset, continued)

(7) Suppose there is R: **C**$\rightarrow$**D** and **C** has all meets. Then R has a left adjoint iff R preserves meets.

Because: ($\Rightarrow$) follows by (6). ($\Leftarrow$) define $L(d) = \sqcap_C\{c' \mid d \leq Rc'\}$. Then $d \leq Rc \Rightarrow Ld = \sqcap_C\{c' \mid d \leq Rc'\} \leq c$. On the other hand: $Ld \leq c \Rightarrow R(\sqcap_C\{c' \mid d \leq Rc'\}) \leq Rc \Rightarrow d \leq \sqcap_C\{Rc' \mid$

$d \leq Rc'\}) \leq Rc$. □

There are several results which generalize the previous example. We present just one of them. Given a functor R: **C**→**D**, where **C** is small and has all limits the following *solution set condition* plays an important role in establishing the existence of a left adjoint.

$\forall d \in \mathbf{D}.\ \exists\ \{(c_i, w_i: d \to Rc_i)\}_{i \in I}$ set.
$\forall c' \in \mathbf{C}.\ \forall f: d \to Rc'.\ \exists\ i.\ \exists\ f': c_i \to c.$
$(f = Rf' \circ w_i)$

Exercise: Show that if **D** is small or R has a left adjoint then the solution set condition is always satisfied.

This can be usefully understood as a weakening of the universal condition in the definition of a universal arrow. Given an object $d \in \mathbf{D}$ we can find a set of objects and maps that are enough to make commute, not in a unique way, every map f: $d \to Rc'$.

Theorem *(Freyd)*
Let **C** be a category with all limits. Then a functor R: **C**→**D** has left adjoint iff R preserves all limits and satisfies the Solution Set Condition.

7. Cartesian Closed Categories

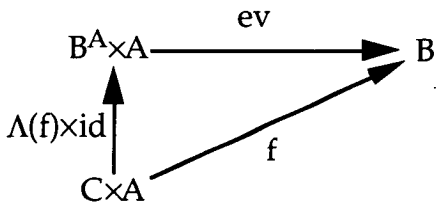
Cartesian closure formalizes the idea of closure of a category under functional space. Chapter 2 provides some intuition for the genesis of the notion and several equivalent formalizations and examples. Here we just present the basic definition and recall that small categories and presheaves are examples of CCC.

Definition *(CCC)*
A category **C** is cartesian closed if it has:

(i) A *terminal object* 1.

(ii) For each A, $B \in \mathbf{C}$ a *product* that we denote with: $A \xleftarrow{\pi^A} A \times B \xrightarrow{\pi^B} B$ such that
$\forall C \in \mathbf{C}.\ \forall f: C \to A.\ \forall g: C \to B.\ \exists! h: C \to A \times B.$
$(\pi^A \circ h = f \ \wedge\ \pi^B \circ h = g)$, (h is often denoted by $\langle f, g \rangle$).

(iii) For each A, $B \in \mathbf{C}$ an *exponent* that we denote with: $ev: B^A \times A \to B$ such that
$\forall C \in \mathbf{C}.\ \forall f: C \times A \to B.\ \exists! h: C \to B^A.\ ev \circ (h \times id) = f$, (h is often denoted by $\Lambda(f)$).



□

Example *(Cat)* The category of small categories and functors is cartesian closed. The exponent object $\mathbf{C} \Rightarrow \mathbf{D}$, is given by the category of functors and natural transformations. Then define:

$ev(F,A)=FA$, $ev(\delta,f)=Gf \circ \delta_A = \delta_B \circ Ff$ (where $\delta: F \to G$, $f: A \to B$).

$\Lambda(F)AB=F(A,B)$, $\Lambda(F)Af=F(id_A,f)$, $\Lambda(F)fB=F(f,id_B)$. □

Example *(Presheaves)* Our next example of a CCC is $\mathbf{Set}^{C^{op}}$ (or $\mathbf{C}^{op} \Rightarrow \mathbf{Set}$), for any **C**. The cartesian structure is built "pointwise", but this does not work for exponents (try to take $(F \Rightarrow G)A=\mathbf{Set}(FA, GA)$, how do you define $F \Rightarrow G$ on arrows?). The solution is to use Yoneda Lemma. For $F, G: \mathbf{C}^{op} \to \mathbf{Set}$, define $F \Rightarrow G$, ev and $\Lambda(\delta)$ (for $\delta: H \times F \to G$) by

$(F \Rightarrow G) = \lambda c.\ Nat[\mathbf{C}[\_,c] \times F, G]$

$ev = \lambda A(\delta,u).\ \delta_A(id_A, u)$

$\Lambda(\delta) = \lambda a u b(f,v).\ \delta_B(Hfu, v)$ (where $u \in HA$, $v \in FB$, $f: B \to A$) . □

Exercise: If **C** is a preorder, we can recover a pointwise definition of $F \Rightarrow G$. Define $\mathbf{C}\lceil A$ as the full subcategory of **C** with objects those B such that $B \leq A$. Given $F: \mathbf{C}^{op} \to \mathbf{Set}$, define $F\lceil A : (\mathbf{C}\lceil A)^{op} \to \mathbf{Set}$ by restriction. Then show: $(F \Rightarrow G)A=\mathbf{Set}(F\lceil A, G\lceil A)$. □

Exercise CCC-0 : Let **C** be a CCC which has an initial object 0. Then show that for any A: (i) $0 \times A \cong 0$, (ii) $\mathbf{C}(A,0) \neq \emptyset \Rightarrow A \cong 0$ (thus $\mathbf{C}(A,0)$ has at most one element). If furthermore **C** has finite limits, show that, for any A, the unique arrow from 0 to A is mono. Hint: $\mathbf{C}(0 \times A,B) \cong \mathbf{C}(0,A \Rightarrow B)$, and consider in particular $B=0 \times A$. Consider also $!^{op} \circ \pi$. Suppose $f: A \to 0$. Then consider $\pi' \circ \langle f,id \rangle$.

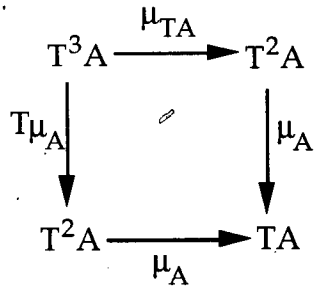
Exercise CCC-$\neg$: Let **C** be a CCC, and 0 be an object such that the natural transformation μ: $\lambda x.x \to \lambda x.(x \Rightarrow 0) \Rightarrow 0$ defined by $\mu=\Lambda(ev \circ \langle \pi', \pi \rangle)$ is iso. Show that 0 is initial and that **C** is a preorder. (This is an important negative fact: there is no nontrivial categorical semantics of classical logic, thinking of 0 as absurdity and $(x \Rightarrow 0) \Rightarrow 0$ as double negation.) Hint: (i) $0 \Rightarrow 0 \cong 1$ (indeed $1 \cong (1 \Rightarrow 0) \Rightarrow 0$, and $1 \Rightarrow A \cong A$, for any A). (ii) for any A: $\mathbf{C}(0,A) \cong \mathbf{C}(0,(A \Rightarrow 0) \Rightarrow 0) \cong \mathbf{C}(0 \times (A \Rightarrow 0),0) \cong \mathbf{C}(A \Rightarrow 0, 0 \Rightarrow 0) \cong \mathbf{C}(A \Rightarrow 0,1)$. (iii) for any A,B: $\mathbf{C}(A,B) \cong \mathbf{C}(A,(B \Rightarrow 0) \Rightarrow 0) \cong \mathbf{C}(A \times (B \Rightarrow 0),0)$.

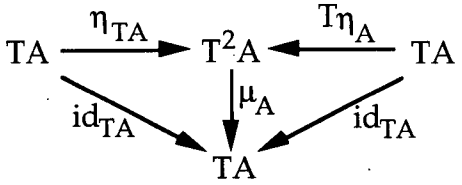
8. Monads

The notion of monad (or triple) is an important category theoretic notion, we refer to Barr&Wells[85] and MacLane[71] for more information and to chapter 4 for several applications of this notion in computer science.

Definition *(monad)*

A monad over a category **C** is a triple (T, η, μ) where $T: \mathbf{C}\to\mathbf{C}$ is a functor, $\eta: Id_C \to T$, $\mu: T^2 \to T$ are natural transformations and the following diagrams commute:

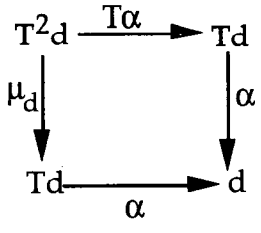


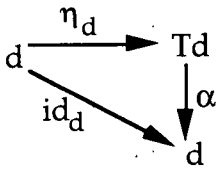


Exercise: Show that if **C** is a poset then a monad can be characterized as a closure, i.e. $T: \mathbf{C}\to\mathbf{C}$ monotone, such that $Id \le T = T\circ T$.

Definition *(category of T-algebras)*

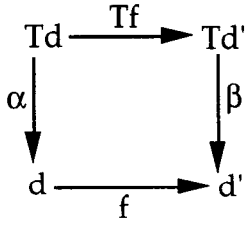
Given a monad (T, η, μ) a T-algebra is a morphism $\alpha: Td\to d$ satisfying the following conditions:





The category **T-alg** has T-algebras as objects and as morphisms

$$\mathbf{T\text{-}alg}[\alpha: Td\to d, \beta: Td'\to d'] \triangleq \{f: d\to d' \mid \beta\circ f = Tf\circ\alpha\}$$



Definition *(Kleisli category)*

Given a monad (T, η, μ) over the category **D**, Kleisli category $\mathbf{K_T}$ is defined as:

$\mathbf{K_T} \triangleq \mathbf{D}$ $\qquad$ $\mathbf{K_T}[d, d'] \triangleq \mathbf{D}[d, Td']$

$id_d \triangleq \eta_d: d\to Td$ $\qquad$ $f\circ g \triangleq \mu_{d''}\circ Tf\circ g$, if $g: d\to d'$, $f: d'\to d''$ in $\mathbf{K_T}$.

Theorem

(1) Every adjunction $(L, R, \eta, \varepsilon)$ gives rise to a monad $T(L \dashv R) \triangleq (RL, \eta, R\varepsilon L)$.

(2) Given a monad (T, η, ε), in the category **D** consider the category of T-algebras, **T-alg**, then we can build an adjunction $(L^T, R^T, \eta^T, \varepsilon^T)$ as follows:

$L^T(d) \triangleq (\mu_d: T^2d \to Td)$ $\quad$ $L^T(f: d \to d') \triangleq T(f)$

$R^T(\alpha: Td \to d) \triangleq d$ $\quad$ $R^T(g: \alpha \to \beta) \triangleq g$

$\eta^T \triangleq \eta$ $\quad$ $\varepsilon^T(\alpha: Td \to d) \triangleq \alpha$.

Moreover the monad induced by this adjunction is again (T, η, ε).

(3) Given a monad (T, η, ε), in the category **D** consider the Kleisli category $\mathbf{K_T}$ then we can build an adjunction $(L^{KT}, R^{KT}, \eta^{KT}, \mu^{KT})$ as follows:

$L^{KT}(d) \triangleq d$ $\quad$ $L^{KT}(f: d \to d') = \eta_{d'} \circ f$

$R^{KT}(d) \triangleq Td$ $\quad$ $R^{KT}(f: d \to Td') \triangleq \mu_{d'} \circ Tf$

$\eta^{KT} \triangleq \eta$ $\quad$ $\varepsilon^{KT}{}_d \triangleq id_{Td}$.

Moreover the monad induced by this adjunction is again (T, η, ε).

We conclude by mentioning the special role played by Kleisli and T-algebras adjunctions among the adjunctions generating a given monad.

Definition ***(Adj(T))***

Given a monad $\mathbf{T}=(T,\eta,\mu)$ the category **Adj(T)** of **T**-adjunctions is defined as:

$Ob_{Adj(T)} = \{(L', R', \eta', \varepsilon') \mid L' \dashv R'$ and $\mathbf{T}(L' \dashv R') \equiv (R'L', \eta', R'\varepsilon'L') = \mathbf{T}.$

$\mathbf{Adj}(\mathbf{T})[(L', R', \eta', \varepsilon'), (L'', R'', \eta'', \varepsilon'')] = \{\Phi$ functor $\mid \Phi \circ L' = L''$, $R'' \circ \Phi = R'\}$.

Fact Given a monad the Kleisli adjunction is initial and the T-algebra adjunction is final in **Adj(T)**.

Analytic Index

Unité de Recherche INRIA Lorraine
Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers Lès Nancy Cedex (France)
Antenne de Metz
Technopôle de Metz 2000 - Cescom - 4, rue Marconi - 57070 Metz (France)

Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 Rennes Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 Grenoble Cedex (France)
Unité de Recherche INRIA Rocquencourt Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex

ISSN 0249 - 0803